Semi-Automated Grading of Novice Java Programs for an Online Grading System

Micheal Alexis Ponniah
Montclair State University

Follow this and additional works at: https://digitalcommons.montclair.edu/etd

Part of the Computer Sciences Commons

Recommended Citation
https://digitalcommons.montclair.edu/etd/1236

This Thesis is brought to you for free and open access by Montclair State University Digital Commons. It has been accepted for inclusion in Theses, Dissertations and Culminating Projects by an authorized administrator of Montclair State University Digital Commons. For more information, please contact digitalcommons@montclair.edu.
Semi – automated grading of Novice Java Programs for an Online Grading System

by

Micheal Alexis Ponniah

A Master’s Thesis Submitted to the Faculty of Montclair State University

In Partial Fulfillment of the Requirements For the Degree of

Master of Science

January 2006

School College of Science and Mathematics
Department Computer Science

Thesis Committee:

John Jenq, Ph.D.
Thesis Sponsor

Dorothy Deremer, Ph.D.
Committee Member

Carl Bredlau, Ph. D.
Committee Member

Dorothy Deremer, Ph.D.
Department Chairperson

12/21/05
(Date)
SEMI-AUTOMATED GRADING OF NOVICE JAVA PROGRAMS FOR AN ONLINE GRADING SYSTEM

by

MICHEAL ALEXIS PONNIAH

A THESIS
Submitted in partial fulfillment of the requirements
For the degree of Master of Science in
The Department of Computer Science in
the Graduate Program of
Montclair State University
January 2006
Semi – Automated Grading of Novice Java Programs for an Online Grading System

Micheal Alexis Ponniah

Department of Computer Science, Montclair State University

Upper Montclair, NJ 07043, USA

Abstract

In recent years, World Wide Web usage for teaching and learning has increased rapidly. At the same time, even with some computerized works involved, the traditional grading of home works have not changed too much. Clearly an online grading system would be a highly desirable addition to the educational tool-kit, particularly if it can provide less costly and more effective outcome. The task of grading student programs is neither simple nor entirely mechanical; rather, it is often a tedious and laborious process that is prone to human error. Recently there are a lot of researches in automating this job. Of course, it is acknowledged that there are a number of aspects that are still not amenable to computer implementation: for example, understanding the comments in a program is still beyond the ability of most advanced Artificial Intelligent techniques. However there are many areas in which an automatic grading system can do at least as well or better than human tutor. While grading a program, the correctness of the program plays a vital role. The correctness of a program in our grading environment is determined using a novel method called comparative correctness (CC). In this research, we proposed semi-automatic ways to grade applications. The method CC is presented and tested. The method can be used to test the correctness of methods in a program. Presented in this report included also an interface program that deals with grading program such as applet. This interface program can be applied to applications as well.
Acknowledgements

I would like to thank my advisor, Dr. John Jenq, for his valuable guidance, indispensable advice and eminent support during this thesis at Montclair State University. I would also like to thank my dissertation committee members Dr. Dorothy Deremer and Dr. Carl Bredlau for their support and benevolence towards the completion.

I express sincere appreciation to Dr. Dorothy Deremer, the Chairperson of the Department of Computer Science at Montclair State University for her kind help. I also wish my thanks to Beverly Macaluso for her unforgettable help during my stay here at Montclair State University.

I would like to thank all the faculty at department of computer science for their support and help.

I would also like to thank Irina and Ramazan in working with me and eventually publishing a paper.

I would also like to thank my family members in constantly supporting me.
Contents

1. Overall Description: ................................................................. 1
2. Scope of the research: .............................................................. 1
3. Related Work: ........................................................................... 2
4. Methodology and Implementation ............................................. 5
   4.1 Methodology: ........................................................................ 5
   4.2 Implementation: ..................................................................... 5
5. Correctness: ................................................................................ 6
   5.1 Comparative Correctness: .................................................... 6
   5.2 Runtime I/O Handler (RIOH): ........................................... 11
6. Tool Implementation: ............................................................... 14
7. General Web Architecture .......................................................... 15
   7.1 The Client Layer ................................................................. 15
   7.2 The Presentation Layer ....................................................... 16
   7.3 The Business Logic Layer .................................................. 17
   7.4 The Data Layer .................................................................... 18
8. Life Cycle Of Servlet: ............................................................... 20
   8.1 Session Tracking: ............................................................... 21
9. Test data and results: ............................................................... 22
10. Manual grading ....................................................................... 28
11. Conclusion and future work: ................................................... 30
12. Appendix A: Visual Tour of the website .................................. 31
13. Appendix B: Comparative Correctness Code .......................... 39
15. References .............................................................................. 60
List of Figures

Figure 5.1.1 Comparative Correctness - Ideal case ........................................ 8
Figure 5.1.2 Comparative Correctness – graph for Ideal case .......................... 9
Figure 5.2.1 Runtime Input Output handler .................................................. 11
Figure 5.2.2 Runtime Input Output handler with single event ....................... 12
Figure 5.2.2 Runtime Input Output handler with multi event ....................... 13
Figure 7.1.1 Three Tire Architecture .......................................................... 18
Figure 8.1.1 Life cycle of servlet .................................................................. 20
Figure 9.1.11 Comparative Correctness screen ............................................. 24
Figure 9.1.1 Test Result ............................................................................... 25
Figure 10.1.1 Manual grading screen ............................................................ 29
1. Overall Description:

This research is part of a larger project already underway at Montclair State University. An online system is being developed that will provide feedback to students in two introductory undergraduate programming courses, regarding their Java coding exercises. The purpose of this larger project is to create an expert system that analyzes the semantics of Java code and provides useful feedback to students as they develop their coding skills.

The task of grading student programs is neither simple nor entirely mechanical; rather, it is often a tedious and laborious process that is prone to human error. Recently there are a lot of researches in automating this job. Of course, it is acknowledged that there are a number of aspects that are still not amenable to computer implementation: for example, understanding the comments in a program is still beyond the ability of most advanced Artificial Intelligent techniques. However there are many areas in which an automatic grading system can do at least as well or better than human tutor [1].

2. Scope of the research:

A) In introductory programming courses more than 75% of the programs written by students require human interaction in terms of feeding inputs or viewing outputs. In designing a grading system it is very important to automate this I/O activity to certain level (since complete automation is tedious, for example, in evaluating a ping-pong game).

B) Moreover, determining the correctness of a program is a research topic itself [1]. “Pattern matching”[1,12], a novel and most widely used technique to check the correctness of a program, checks the generated output against the desired output given by the grader. But this technique becomes void when we talk about testing an anonymous program without a specified desired output. For example, to determine the correctness of a program which takes 2 integers as inputs and generates 1 integer as output is not
possible using pattern matching without a desired output specified. Here we introducing a new technique called Comparative Correctness (CC).

3. Related Work:

There is a plethora of work in regard to automatic grading of programs. Searches in the ACM digital libraries with the search term “automatic grading of java program” produced over 200 unique articles. Additional searches using “static analysis java”, “automatic grading system” produced about 250 and more articles. I reviewed abstracts of most of these articles and discovered several that may provide interesting approaches for the development of the overall project.

The work of David Jackson & Michelle Usher [1] appears to be most relevant to my thesis. In this paper the author uses cyclomatic complexity of a program, as defined by McCabe [2], which is not used in our semi-automated system. The grading approach of ASSYST[1] involves a checklist such as correctness, efficiency, style, complexity, and test data. ASSYST is a mainframe centric application and is not deployed in a distributed environment. DASGS[18] is based on the three-tier client/server architecture and utilizes CORBA model. The only weakness of DASGS was the time required to download the Java applets and the object request broker (ORB) middleware from the Web server during runtime. The DASGS was replaced by the WASGS that leveraged middleware, since the WASGS was based on an advanced application server within three-tier client server architecture.

Yu-Liang Chi, in his paper improving the evaluation of programming courses, compares two types of distributed evaluation system for distance education. First type is non-interactive. Once the user logs in he/she will see a list of questions in a static HTML page. Once completing the test, the results are uploaded to a server where its graded. The main advantage of this system is very less network traffic and the performance. Second type is completely interactive. Once the student has logged in he/she attempts first question and the second question depends on the result of the first question. The main
disadvantage of this system is the very high network traffic. For every question the client should communicate with the server and the server should respond appropriately. The advantage being highly interactive and adaptive to different students. Here Yu-Liang Chi, proposes a method combing the both the techniques. He designs a system that will allow the students to download a "question package". Once the user downloads and installs it then this environment acts as a server and provides an interactive experience. At the end of the test, all the results are transferred to the server where it’s further processed.

A similar work by David W. Juedes [3] describes about a web based grading system for a data structures class. The web-based grading system consists of electronic submission and retrieval programs, automated testing and web-based evaluation software, web-based code annotation software, and web-based report generation software. The author designs the software with two objectives, electronic grading and delivery as the primary objective and course based assessment as a secondary objective. In addition, the software provides the means to capture electronic examples of student work for assessment and accreditation purposes and the means to create electronic portfolios for individual students. The author has tested this software in a sophomore-level course on data structures. The software system provided a means to examine detailed information on the overall performance of students on programming projects. In particular, the software recorded information concerning the performance of student projects on test cases and recorded the frequency and type of design and documentation errors. This information was used to evaluate the overall performance of students and to identify common errors that merited additional course time and review.

Derek S. Morris [4] describes an automatic system to grade java programs according to a solution submitted by the instructor. Java reflection classes are extensively used to test each method or class separately. The author describes a suite of programs that performs the automatic grading of Java programming assignments. This suite has been used in the Introduction to Computer Science course at Rutgers University since the Spring 2000 semester where it grades some 400 -600 weekly assignments. The time to grade the
submissions is a matter of minutes on a 500MHZ PC. The main disadvantage of this suite is to create assignment specific evaluation software which takes in the order of 8 hours. The programming system derives its ability to grade a program in the presence of errors using the Java Reflection classes, the Java inheritance mechanism, and Perl Regular Expressions. The Reflection Classes allow a student program's methods to be individually found and executed independent of how they are actually named by the student. The Java inheritance mechanism allows a flawed student method to be overridden with a known good method so the evaluation may proceed. The Regular Expressions allow the student's program output to be gleaned for information even if the output does not adhere to specified format. The Regular Expressions are also used to scan the Student's source code for desirable or undesirable coding patterns. In addition, the systems programming aspects of Perl are used to extract the student programs from the environment where they are submitted and to construct the environment in which they are compiled, executed, and evaluated.

Sykes [5] describes a prototype for a system that will involve a small subset of the Java language. Here the author develops a Java Intelligent Tutoring System (JITS). JITS helps the student to learn where they commit mistakes. It uses a knowledge base to store information.

It's interesting to note that there are more researches in E-learning such as C.J.Butz, S.Hua, R.B.Maguire [6], Gerhard Weber [7], Karahan, Albayrak, Yagmurulu [8] addressing on intelligent and adaptive system for computer programming. In this [8] paper the author discusses the feasibility of implementing Distributed Artificially Intelligent Teaching System – DAI-TS based on the multi-agent architecture approach, aiming at the achievement of human resources qualification through Virtual training.

The work of Jon Palmer, Robert Williams and Helinz Dreher [9] and Marti A. Hearst [10] is more open ended and is addressing the automatic grading on essay’s which will be more relevant when we extend our system to wider domain.
David Wolff[11] in his paper describes about a tool for managing the submission of student work similar to that of blackboard[13].

4. Methodology and Implementation

4.1 Methodology:

The present effort seeks to investigate the correctness of a program using Comparative Correctness and to implement a tool that makes the grading process much easier and faster. Determining the correctness of a program is a research topic itself [1] and is extremely difficult. Therefore, a number of important assumptions and constraints were imposed in the present case in order to define a meaningful scope for this research.

1) Since java is used extensively in introductory computer courses, we concentrate in grading java programs. But it can be easily extended to other programming languages.
2) Hereinafter such programs will be referred as ‘novice code’, ‘novice program’ or ‘Events’.
3) It is assumed that the novice code under investigation has been successfully compiled.
4) The novice program has at least one method with a return type.
5) Each method has at least one but not more than three input parameters.
6) Parameters to a method is limited to primitive types only. Graphical User Interfaces(GUI) are not included (for example, applets, swings)

4.2 Implementation:

Our current research approach focuses on the development of

(A) A Comparative correctness system called, Runtime I/O Handler (RIOH) that has two features:
1) Provides input to a program that expects input from the command prompt from a text file that contains different input values under which a program has to be tested.
2) Stores output to a file which is further used for report and evaluation.
Initially we will consider only numeric inputs of data type int, double and float under command prompt environment. Future work will address other data types and GUI environment(Applets, swings etc.).

(B) A Graphical User Interface (GUI) with Adaptive and graphical module.

5. Correctness:

While grading a student’s solution more attention is paid in evaluating the solution to check if it produces the desired output.

A method used by ASSYST[5] which involves treating the analysis of output as a pattern matching problem is extended and used to evaluate the correctness of a program. Pattern matching states as follows,

For any given event, its outcome is matched with a predefined desired outcome. If both matches then the solution is considered to be correct. Otherwise the solution is wrong.

Let,

$E = \text{set of unique events} = \{E_1, E_2, E_3\}$

$I = \text{set of inputs} = \{i_1, i_2, i_3\}$

$O = \text{set of outcomes by } E_1, E_2, E_3 = \{o_1, o_2, o_3\}$

$D = \text{set of predefined outcomes} = \{d_1, d_2, d_3\}$

For any input (say, $i_j$), if the outcome(say, $o_j$) produced by an event (say, $E_i$) matches the desired outcome $d_j$ then the event $E_j$ is considered correct.

This method could be extended to our specific grading problem. By extending it we are proposing a new method namely Comparative Correctness.

5.1 Comparative Correctness:

Comparative Correctness is a variation of the pattern matching technique mentioned above. We start with a pool of similar anonymous programs (called events) which are supposed to generate the same set of output for the same set of inputs.
Let,

\[ E = \text{set of events} = \{E_1, E_2, E_3\} \]
\[ I = \text{set of inputs} = \{i_1, i_2, i_3\} \]
\[ O = \text{set of outputs} = \{o_1, o_2, o_3\} \]

For any input (say, \(i_j\)), if an event (say, \(E_i\)) generates an output (say, \(o_j\)) then all the other events in the same pool should generate the same output \(o_j\) for the same input \(i_j\).

Output, \(O\), is a function of Input, \(I\), and the Event, \(E\).

\[ O = f(I, E) \]

If, \(I\), remains the same for all events in the same pool then output for different events will remain the same.

**Ideal situation:**

Ideal situation is possible when all the events in a pool produce the same output for the same input. The Figure 5.1.1 below explains the ideal case involved; input \(I\) fed to all the events generates the same output \(O\).
Figure 5.1.1 Comparative Correctness - Ideal case

Figure 5.12 shows ideal case for input I with 10 events that generate an output value of 10. Drawing a graph denoting the behavior of comparative correctness make the concept clearer.
Actual situation:

The output of an event depends on the input and the nature of the event. The nature of an event is a function of the difficulty of an event and the ease to test the events output manually.

\[ O = f (I, E) \]
\[ E = f (d_E, e_E) \]

Where,

\( E \) – the event under investigation.
\( O \) – output of the Event.
\( I \) – output to the Event.
\( d_E \) – difficulty of the Event.
\( e_E \) – easiness to manually test the output.
Let's consider two examples to explain the relationship between an event, difficulty and easiness to manually test the output.

**Example: 1**

Let's assume an event that takes in two parameters as input and returns one output value. The following is an example for such an event written in java that finds the sum of two numbers and returns the value.

```java
public int addition(int a, int b) {
    return a + b;
}
```

The difficulty of such an event is fairly less for an author and it's much easier to test the output manually. So when such events are under investigation the behavior is likely to be close to the ideal case.

**Example: 2**

Let's assume another event that takes in one parameters as input and returns one output value. The following is an example for such an event written in java which finds if a given number is a prime number or not. It returns a Boolean value.

```java
public boolean isPrime(int x) {
    if (x < 2) return false;
    for (int i = 2; i < x; i++) {
        if (x % i == 0) return false;
    }
    return true;
}
```

The difficulty of such an event is comparatively more for an author and it's difficult to test the output manually. So when such events are under investigation the behavior is not likely to be close to the ideal case.
In an actual testing environment the ideal case is rare. So, for different values of Input, \( I \), we record different values of Output, \( O \), for every event, \( E \). The correctness of an event, \( E \) is determined by the comparative analysis of its output with the output of all the other events in the pool for the same set of inputs.

### 5.2 Runtime I/O Handler (RIOH):

In order to effectively test the proposed comparative correctness technique, we developed an environment called Runtime I/O Handler (RIOH). RIOH feeds input for the programs from a text file automatically during runtime and stores the generated output in another text file. The inputs to any given event are predetermined and are stored in a text file.

Java reflection API is extensively used to get a handle over the methods of an event at runtime. The inputs are fed from the file to these methods and the return value is saved in the output file. This output file is further processed for comparison.

Let's take a simple event which calculates the sum of two numbers. It takes two integers as input and returns one integer as an output.

![Figure 5.2.1 Runtime Input Output handler](image-url)
public class Sum{
    private int x,y,sum;
    private int sum(int a, int b){
        x = a;
        y = b;
        sum = x + y;
        return sum;
    }
}

When this event is run under the RIOH,

Figure 5.2.2 Runtime Input Output handler with single event
The inputs (2, 2), (7, 3) and (23, 7) are fed from the input file and the outputs 4, 10, 30 respectively are stored in the output. The Figure 5.2.2 represents the behavior of RIOH for a single event in the pool. Figure 5.2.3 shows RIOH when multiple events are presented.

In deciding the correctness of a program using comparative correctness, it's important to note that the impact of certain inputs on an event does not change the behavior (output) of the event. In the above example using the inputs (2, 2) and output 4, we cannot conclude that the behavior of the event is correct and that it does addition of (2,2). Because for the same input (2, 2) the following two events will produce the same output.

Figure 5.2.2 Runtime Input Output handler with multi event
Event 1:

```java
public class Prod{
    private int x,y,prod;
    private int prod(int a , int b){
        x = a;
        y = b;
        prod = x * y;
        return prod;
    }
}
```

Event 2:

```java
public class Power{
    private int x,y,power = 1;
    private int pow(int a , int b){
        x = a;
        y = b;
        for (int i = 0; i < y; i ++)
            power = power * x;
        return power;
    }
}
```

There are two approach to this problem

1) To avoid using such inputs in test cases
2) To use a knowledge base of semantics.

The first approach allows us to consider the events as a black box where as the second approach forces us to analyze the semantics of the event.

6. Tool Implementation:

In recent years, WWW usage for teaching and learning has increased rapidly. At the same time, the traditional paper based grading of home works have not changed too much.
Recently there have been many researches on automated and intelligent grading. Clearly an online grading system would be a highly desirable addition to the educational tool-kit, particularly if it can provide less costly and more effective outcome.

In this research a Web-based Grading System (WGS) was used. The system has several components including online assignment submission, an interface for students to read, review and assess their assignments, check their graded assignments. WGS is a Web-based application and can interact with the user through a Web browser. Javascript is used for client side validation. Apache Tomcat is used as the web server. Servlet is used for server side programming. MySQL serves as the database backend. Structured Query Language (SQL) is used for composing the queries to access the database.

7. General Web Architecture

7.1 The Client Layer

The client layer of a web application is implemented as a web browser running on the user's client machine. The main job of this layer is to display data and let the user enter or update data. To build this layer there are two general approaches:

1) **Static HTML client**: As the name indicates the client layer is used only to display and get data from the server. All data processing is placed in the middle tier. When the user submits the data, all the validation is done in the middle tier and any errors are posted back to the client as a new page. This approach creates more traffic and is vulnerable to security issues.

2) **Dynamic HTML client**: In this approach the client acts as a gateway to inputs. We could make the HTML page not to allow certain types of inputs. For example, for a phone numbers input we could check if all the inputs are numbers and ensure that the number of digits typed is valid. If any of these are wrong then we could alert the user to re-enter the data. The main advantage of this approach is that, we don't have to communicate with the database every time the user typed wrong information. This will greatly decrease the network traffic.
The dumb client approach tends to be more cumbersome for end-users because it must go back-and-forth to the server for the most basic operation. Also, because lists are not built dynamically, it is easier for the user to inadvertently specify invalid combinations of inputs (and only discover the error on submission). The first argument in favour of the dumb client approach is that it tends to work with earlier versions of browsers (including non-mainstream browsers). As long as the browser understands HTML, it will generally work with the dumb client approach. The second argument in favour of the dumb client approach is that it provides a better separation of business logic (which should be kept in the business logic tier) and presentation (which should be limited to presenting the data). Including Dynamic HTML and JavaScript in the Presentation (so it can run on the client) mixes the tiers.

7.2 The Presentation Layer

Presentation layer serves as the middle layer between the client and the data layer. It decodes the data from the client. Typically data entered by the user that involves some process. For example, user name and password. Once these data are in the presentation layer, they are compared with the data in the data layer. To do so, presentation layer requests the data layer for these fields and checks if they match.

The Presentation layer can be built by a number of different tools. In early websites CGI or Common Gateway Interface programs were used to design the presentation layer. Now for modern websites developers prefer

- the Microsoft solution using Active Server Pages (ASP) which may be generated by Visual InterDev
- the Java solution using some combination of Servlets and JavaServer Pages (JSP)

These tools help developers in embedding dynamic content inside other static HTML in the webpage. They also provide tools that make it easy to parse a webpage coming back from the client to get the user-entered information.
The presentation layer is usually inside a Web Server (like Microsoft IIS, Apache WebServer, Weblogic, IBM Websphere, etc.) A single web server could be configured to serve more than one website. By changing the configuration file, web server could handle several requests. Usually in a production environment, more than one web servers are connected together to serve a single web site. This is usually referred to as web farms. If one server goes down then another server in the web farm will take care of it. Similarly if the traffic to one server is more then the traffic is divided and distributed evenly to all other servers.

7.3 The Business Logic Layer

Application logic is usually implemented in the business logic layer. Business logic includes:

- performing all required calculations and validations
- managing workflow (including keeping track of session data)
- managing all data access for the presentation tier

Business logic usually implements the data processing involved. For the Microsoft solution COM objects are built using Visual Basic or C++. For Java solution Enterprise Java Beans (EJB) are built using Java. Language-independent CORBA objects can also be built and easily accessed with a Java Presentation Tier.

The business logic layer is generally implemented inside an Application Server (like Microsoft MTS, Oracle Application Server, IBM Websphere, etc.) The Application Server generally automates a number of services like transactions, security, persistence/connection pooling, messaging and name services. Isolating the business logic from these “house-keeping” activities allowing the developer to focus on building application logic while application server vendors differentiate their products based on manageability, security, reliability, scalability and tools support.

Typically for small applications both presentation and business layer are combined together.
7.4 The Data Layer

The data layer is responsible for managing the data. In the most case, a data layer may simply be a modern relational database. However, it may include data access procedures to other data sources like hierarchical databases, legacy flat files, XML files etc. The job of the data layer is to provide the business logic layer with required data when needed and to store data when requested. Data layer does not interact directly with the client layer. This makes the system more stable from the clients data. Since all the data entered should pass through the presentation/business layer, the data entered by client are further screened and made sure that the data entering the data layer is the exact data that was expected. Figure 7.1.1 is a three tire architecture for web application.

![Figure 7.1.1 Three Tire Architecture.](image-url)
In tier I, the user interacts with the server using a web browser. Javascript is used to validate user input. The client requests the web server for HTML page and the server responds to it. A web browser places a http request and the webserver returns with a http respond.

In tier II servlet is used for server side programming. Tomcat serves as the servlet container. Dynamic HTML pages are returned to the browser by the servlet. Servlet communicates with the database through JDBC to authenticate the user, store and retrieve data. This tier is also called as application layer.

In tier III, MySQL is used as the database server. It receives requests from the servlet in the form of SQL queries, executes them and returns the dataset back to the servlet.

Javascript is used to do validation on the client side. That is, when the client is filling a form, at that time he/she are not allowed to enter invalid data and forces the user to enter data in the required/compulsory fields before submitting the data (to the server).

Servlets are used in the middle tier. They are the Server side programs written in Java. They reside inside the default directory of Web Server and are executed by the Web Server. The client will interact with HTML page, fill the data and submit it. The Browser will send the data as a request to the Servlet program (the name we have mentioned in the form tag’s action attribute) and store the data in the session or in the database as require.

The Advantages of Servlet are

a) Servlets are persistent. Servlet are loaded only once by the Web Server and can maintain services between requests. On the other hand, CGI (Common Gateway Interface) scripts must be loaded and executed by Web Server each time a request is made to it.

b) Servlet are fast. They offer better performance since they need to be loaded once.

c) Servlet are platform independent as they are written in Java. Because the server available to run the servlet program are actually platform independent and can be installed on any operating system e.g. UNIX, Linux, Windows etc.

d) Servlet are extensible. All the benefits of Java can be brought into our Servlet.

e) Servlet are Secure. The only way to gain access to a file is through Server. If server is protected, the Servlet is protected as well.
f) Servlet can be used with a variety of client i.e. Browser, applet etc.

Apache Tomcat:

Apache Tomcat is the open source servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages.

a) Server Side Include (.shtml) program can be executed by this servlet.
b) Servlet chaining can be done.
c) Most important point, it always gives you the updated response from the servlet program even if the server is running and you change the servlet program, compile it and copy it in the default directory of the server (i.e. servlets). No need to restart the server.
d) It also provides a default directory to copy the html files and provide the security to them.

8. Life Cycle Of Servlet:

Figure 8.1.1 shows the life cycle of a servlet.

```java
public void init(ServletConfig config) { }

public void service(HttpServletRequest request, HttpServletResponse response) { }

public void destroy() { }
```

Figure 8.1.1 Life cycle of servlet

a) init(ServletConfig config): This method is called before the servlet handles its first request. This is the first method to execute in the lifecycle of the servlet. It is used to
initialization the servlet by creating and loading objects that are used by the servlet in handling of it request.

The ServletConfig object is passed as a parameter to supply servlet information about its initialized parameter. These parameters are given to the servlet itself and are not associated with any single request.

b) service(HttpServletRequest request, HttpServletResponse response) :
This method is executed as many times as the client makes request. This method handles the request coming from client and send response back to the client.
HttpServletRequest is an interface which is use to get the information and data related to client request.
HttpServletResponse is an interface used to send the response back to the client.
Different service methods are available to handle request and response doGet(), doPost(), service() (This method will also have these two interface as an argument)etc.

c) destroy(): It is called to free all the memory space used to initialize all the object for the servlet for execution and they are garbage collected.

8.1 Session Tracking:

In servlet, sessions are created and maintained explicitly by calling the HttpServletRequest method getSession(). The servlet engine manages a table of sessions, each identified by a unique string, the session key. The response message is used to send the session key to the client, which is received back with future client request. So that the server can identify the client and retrieve the HttpSession object associated with that client request. If a client does not continue session within a specified time the server will expire the session and delete all data associated with that session. The session key is then invalidated, so the server does not accept the next request coming from the client, which may still include the old session key.
There are several advantages of Session Tracking:

a) Session acts as a good medium to store the data which is often required as the client visit different pages of a website. Such as userid and password.

b) Popularity ratio of a website can also be found out by session tracking.

c) User can be stopped from visiting the pages randomly.

9. Test data and results:

The comparative correctness technique, implemented using Runtime I/O Handler (RIOH), was tested on a group of students taking CMPT 183-02 at the Computer Science department of Montclair State University.

Specification:

Create a class that contains a method that computes the modules of 2 numbers. This method takes two parameters as input and returns one parameter as the output. Typical solution to this problem would be as follows.

/ **
* Constructor for a Mod.
* *
* @author
* @version 2005.4.10
* /
public class Mod
{
    // Member variables
    private int x;
    private int y;
    / **
     * Constructor
    
}
public Mod()
{
    // the instance variables of x and y
    x = 0;
    y = 0;
}

/**
 * Method to compute the modules of two given numbers
 */
public int computeMod(int x, int y)
{
    // The mod of x and y.
    return (x % y);
}

Results:
Number of students participated : 9
Number of events meeting the specification : 7
Number of events not meeting the specification : 2

Running the programs for comparative correctness gives the following page,
<table>
<thead>
<tr>
<th>Student Id</th>
<th>File Name</th>
<th>Output</th>
</tr>
</thead>
</table>
| 1          | now.java  | Mod has 1 methods. They are: public int Mod.computeMod(int, int) a= computeMod  
For the inputs 32, 6 result= 2 |
| 2          | testing.java | Mod has 1 methods. They are: public int Mod.computeMod(int, int) a= computeMod  
For the inputs 32, 6 result= 2 |
| 3          | testing.java | Mod has 1 methods. They are: public int Mod.computeMod(int, int) a= computeMod  
For the inputs 32, 6 result= 2 |
| 4          | He.java   | Mod has 1 methods. They are: public int Mod.computeMod(int, int) a= computeMod  
For the inputs 32, 6 result= 2 |
| 5          | Mod.java  | Mod has 1 methods. They are: public int Mod.computeMod(int, int) a= computeMod  
For the inputs 32, 6 result= 2 |
| 6          | Mod.java  | Mod has 1 methods. They are: public int Mod.computeMod(int, int) a= computeMod  
For the inputs 32, 6 result= 2 |
| 7          | Mod.java  | Mod has 1 methods. They are: public int Mod.computeMod(int, int) a= computeMod  
For the inputs 32, 6 result= 2 |
| 8          | Mod.java  | Mod has 1 methods. They are: public int Mod.computeMod(int, int) a= computeMod  
For the inputs 32, 6 result= 2 |
| 9          | Mod.java  | Mod has 1 methods. They are: public int Mod.computeMod(int, int) a= computeMod  
For the inputs 32, 6 result= 2 |
| 10         | Mod.java  | Mod has 1 methods. They are: public int Mod.computeMod(int, int) a= computeMod  
For the inputs 32, 6 result= 2 |
| 11         | Mod.java  | Mod has 1 methods. They are: public int Mod.computeMod(int, int) a= computeMod  
For the inputs 32, 6 result= 2 |

**Figure 9.1.1 Comparative Correctness screen**

Figure 9.1.1 shows the test run results on the browser screen.

Figure 9.1.2 summarize the result in tabular form.
<table>
<thead>
<tr>
<th>Input 1</th>
<th>Input 2</th>
<th>E1</th>
<th>E2</th>
<th>E3</th>
<th>E4</th>
<th>E5</th>
<th>E6</th>
<th>E7</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>

**Figure 9.1.2 Test Result**

The above results are all consistent. All the events that met the specification produce the same behavior for the same set of inputs.
Figure 9.1.3, Figure 9.1.4 and Figure 9.1.5 are depicted from the data above.

**Figure 9.1.3 Graph for input case I**

**Figure 9.1.4 Graph for input case II**
Figure 9.1.5 Graph for input case III

Thus by comparative correctness we could conclude that, for the above inputs the behavior of the events are not wrong. It’s interesting to note that the programs that did not meet the specification are of the same format. Following is an example,

```java
/**
 * this class contains a method call computeMod
 * @author
 * date: 10/4/05
 */
public class Mod
{
    private int a;
    private int b;
    private int remainder;
    private int result;
    /**
     * Constructor for objects of class Mod
     */
    public Mod()
    {
    }
```
```java
public void computeMod(int x, int y) {
    a = x;
    b = y;
    result = a/b;
    remainder = a%b;
    System.out.println("The result is: "+result+" The remainder is: "+remainder);
}
```

Though these events are not wrong while calculating the modules of two numbers, they don't meet the specification. Addressing this problem is out of the scope of this research. Comparative correctness proves to be highly efficient in a grading environment.

10. Manual grading:

In case of grading an applet our system is not capable of grading it automatically. So we have designed a user interface that can aid the grader in grading the program. This interface maintains a list of common errors committed by students and helps the grader to grade it faster. Figure 10.1.1 gives the screen shot of the system. For detail implementation see appendix C.
Figure 10.1.1 Manual grading screen
11. Conclusion and future work:

Automatic grading of any given program is tedious. But it is possible to automate the grading process with certain constraint. In this thesis we have developed a system to grade java programs automatically with few conditions. Determining the correctness of a given program plays a vital role in a grading environment.

In this thesis, we propose an approach, called Comparative Correctness, to find the correctness of a program utilizing the fact that the student assignment program’s are easy to test manually and student’s knowledge in a class room is highly diverse.

The Comparative correctness technique was tested in CMPT 183, an introductory course for java programming. The test results for Comparative correctness are highly convincing. The current system grades program that involves primitive data types. And we do not address the automatic grading of java applets. Future works will address such types. Extending the current system to a web service application would be highly efficient since comparative correctness is independent of the specification of a program.
12. Appendix A : Visual Tour of the website
(existed before this research was started)

(1) Homepage
Filename: log.htm

It is the first page which opens whenever a client visits the website. From this page the student or faculty should login, by providing the userid, password and have to mention their type i.e. either they are faculty or student by selecting the corresponding radio button.

From this html page, the data is passed to the servlet program login. It also contains 2 hyperlinks to the page studentre.htm and changepass.htm.

(2) Login Servlet
Java File: login.java

This file gets the data sent by the client from log.htm page and checks whether the information provided by the client is correct or not by comparing the values provided by the user and the values present inside the database.

If a client select student radio button, than it will compare the ID and Password from the student_details table. Else if a client select faculty radio button, than comparison will take place from staff_detail table. If the client is a valid user, then his/her id and password are stored in a session and the next page is sent as a response to a client. For the student data.htm page is sent and for faculty faculty.htm is sent.

But if the user password is only incorrect then re-enter.htm is sent to the client. And if both user id and password are wrong, then a warning message is displayed to the client and a link is provide to the log.htm page , to go back to the main screen.
(3) Re-enter Password Screen
File name: re-enter.htm

Sorry you have entered wrong password, please enter the correct password

Enter your Password :

LogOn  Reset

It will only ask the user to re-enter his/her password and will send the data to the servlet program re-enter.

This page is send as a response to the user by the servlet program login. By redirecting the client response by response.sendRedirect(String URL) method.

(4) Re-enter Servlet
Java File: reenter.java

This program will first of all retrieve the values that have been stored inside the session i.e. userid and his/her type. This program checks again the password entered by the user and the password in the database related to that particular id is equal or not.

If user id and password are equal then it will send the data.htm page as a response to a student and faculty.htm to a faculty.

(5) Assignment Submitting Screen
File name: data.htm

SUBMITTING ASSIGNMENT

Assignment No. : [Assignment-1]  Enter Your File Name : [filename]
Check Result

Now from this page the student can submit their assignments. The student has to select the assignment name from the combo box and have to give the filename by which he/she wants to save the program.

The filename should have "java" extension. Scripting is done to prevent the student from entering invalid file name.
It also provide a link to the servlet program showmarks so that the student can see their marks if the faculty has evaluated there assignments.

From this page the request is passed to a servlet program getassignment.

(6) Getting Assignment Data Servlet
Java File: getassignment.java

This program will get the data submit by the student from the page data.htm. First of all it will generate a temporary java file in which it will write the java program coding submitted by the student. Then it will compile it. If compilation is successful it run the program and will save the output in the table student_assignment.

If compilation error is there, the error message is stored in the table. Further more the student id, filename, file data, date, time, assignment name, checked or not etc information is stored inside this table student_assignment.

(7) Assignment Evaluation Screen
File name: faculty.htm

From this page the faculty has to only select the assignment name from the combo box, related to which faculty wants to see the list of file submitted by the student. From this page the request is passed to the servlet program evaluate (due to scripting).

(8) Assignment Evaluation Servlet
Java File: evaluate.java
You have to select the assignment name from the drop down list and after that it will display all the information about the file submitted by the student for that particular assignment. If you want to see the output of a particular file just select the radio button from the row of the corresponding file and press the "Out-Put" button.

It displays all the information regarding the file submitted by the student for a particular assignment. Suppose the faculty has select “Assignment-1” from the combo box, this program will accept this request and it will pass a select query in to the database for selecting all the information of the files, which belongs to “Assignment-1”.

Now if the faculty wants to see the output of any of the file he/she has to select the radio button corresponding to each file data and press the “Show Out Put” button. From here the request is send to the servlet program execution.

NOTE: The faculty can only see output of any one file at a time.

And if the faculty wants to delete a particular file from the database table he/she has to press the delete button. By selecting a radio button corresponding to each file data. Then that file data is going to be deleted permanently from the table student_assignment.

(9) Showing Student Program’s Out Put Servlet
Java File: execution.java

The output of the program: "hello.java" is :

Hello from system 12-6

- Back To Assignment Evaluation Screen.
This program gets the request of which file the faculty want to see the output. It stores the information regarding the file i.e. assignment name, student id, filename, date of submitting, and time in the session (i.e. `session.putValue("fileinformation", v);` where `fileinformation` is session key name and `v` is Vector class object. Vector is a dynamic array).

After this it shows the output of the selected file by executing a select query in the database from the table `student_assignment`. And show the output, but if compilation error is there, then that error message is displayed.

It provides the form by which the faculty can give marks/grade to the program submitted by the student. And this request is passed to the servlet program `storemarks`.

It also provides a hyperlink to servlet program `showcode` so that the faculty can see the coding to the student program.

(10) Marks Storing Servlet

**Java File:** `storemarks.java`

It stores the marks given by the faculty after evaluating the student program in the database table `student_result` (i.e. student id, assignment name, filename and mark/grade).

Further more it updated the student_assignment table by putting the value "yes" in the column named `checked`. So that the particular file should not be listed again in the list of the file as it has been evaluate.

It provides a link to the servlet program `evaluate` so that the faculty can check some more programs and go back, if wants. URL rewriting is done in the URL of this link, so that it will again display the same list of the files, belong to that assignment as that of evaluated by the faculty.

(NOTE: But not that file which has been already evaluated).

(11) Student Programs Code Showing Servlet

**Java File:** `showcode.java`

It shows the coding of the file selected by the faculty, by selecting the data from the table `student_assignment`. It gets the information regarding which file data to be displayed from the session (i.e. session key `fileinformation`).

```
1: class hello
2: {
3:     public static void main(String arg[]) {
4:         System.out.println("Hello from system 12-6");
5:     }
6: }
7: }
```

NOTE: It is not compulsory right now to evaluate the marks. It dependents on you.

<table>
<thead>
<tr>
<th>Student Id</th>
<th>[stu01]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment Name</td>
<td>Assignment-1</td>
</tr>
<tr>
<td>Program Name</td>
<td>hello.java</td>
</tr>
<tr>
<td>Marks/Grade</td>
<td></td>
</tr>
</tbody>
</table>

- Back To Assignment Evaluation Screen.
And also provide a form to evaluate the program submitted by the student. This form again sends the request to the servlet program `storemarks`.

(12) Student Registration Screen  
File name: studentre.htm

![Student Registration Form](image)

NOTE:  
- Your password should be at least 5 character long.  
- Mention your address also.

This is a form for the students to register themselves to use the website to submit their assignment online. So for that they have to get their user id by submitting this form.

The students have to give the required and applicable information about them then only they can be registered. And this request is sent to the servlet program `newuser`.

To prevent the user from entering invalid and randomly filling the form, scripting is done.

(13) Registering New User Servlet  
Java File: `newuser.java`

It gets the request of the student to register them from the `studentre.htm` page. It inserts the data of the student in the database table `student_detail`.

If the registration is successful a user id is going to be generated automatically by the server. The user has to remember his/her user id and password and use them for login successfully.

And a link is provided to connect back to the home page `log.htm`, so that the user is logged on.

(14) Password Changing Screen  
File Name: `changepass.htm`
This page will help both the faculty and the student to change their password. Here they have to provide their user id, old password, new password and also their type either they are student or faculty.

From this page the request is sent to the servlet program `password`.

(15) Password Change Servlet  
Java File: `password.java`

This program gets the request and compares the values provided by the user and the values present in the database. It compares the values from the database table `student_detail`. And updated the table column `password` and put the new value provided by the user.

And a link is provided to connect back to the home page `log.htm`, so that the user can login again.

(16) Marks Showing Servlet  
Java File: `showmarks.java`

This program is called by the link named “Result” in the `data.htm` file. It gets the user id, which is stored in the session when the user logged in first from the home page `log.htm` and send the request to the servlet program login (i.e. `sess.putValue("userin",v);`).

It would only display the marks of that assignment which have been evaluated and not which are remaining from the database table `student_result` and `student_detail`. 
It will display user id, student name and assignment name, filename, marks/grade in a table format. It provides a back link again to the page *data.htm*, so that the student can submit more assignments.

**NOTE:**

- There is no registration form for the faculty to register themselves online due to security reason. Because students may also misuse that form by register themselves as a faculty and evaluate assignment submitted by the student or might delete them.

- So, faculty has to get their id from the website administrator, who is maintaining the server. And if the faculty wants to change his/her password, they can change it from the website.

- This website is platform independent as it is created in Java. Because the server available for running the servlet program are platform independent and can be install on any operating system, such as Window 95, 98 or Unix or Linux. And the servlet programs are copied in the default directory of the server and are executed by the server as the client sends the request.

- Servlet program are actually the “.java” files and we have to compile them. When we compile a java files then a corresponding “.class” files is generated. And to execute the servlet program we require this class files.
import java.io.*;
import java.net.*;
import java.lang.reflect.*;
import java.sql.*;
import java.nio.channels.FileChannel;
import java.nio.MappedByteBuffer;
//using reflection API
public class ComparativeCorrectness {
    static Class c = null;
    //create a file object
    static Fileln Fin = new Fileln();
    //read from the file
    static int a=Fin.getInt();
    static int b=Fin.getInt();
    public static void main(String args[]){
        try {
            String methName="";
            //get the name of the class from command line argument
            c = Class.forName(args[0]);
            Object cobj = null;
            //create an object of the class
            cobj = c.newInstance();
            //variable to hold the # methods and parameter types
            Class partypes[] = new Class[2];
            //get the methods in this class
            Method m[] = c.getDeclaredMethods();
            //print the details of the method
            System.out.println(args[0]+" has "+m.length+" methods. They are: ");
            for(int mm = 0;mm<m.length;mm++){
                System.out.println(m[mm]);
            }
            //get the parameter of the first method
            partypes = m[0].getParameterTypes();
            Integer[] par = new Integer[partypes.length];
            String prin="For the inputs "+a++", "+b++" ";
            for(int p = 0;p<partypes.length;p++){
                if(p==0)
                    par[p] = new Integer(a);
                if(p==1)
                    par[p] = new Integer(b);
            }
methName = m[0].toString();

// get the name of the method
methName

methName.substring(methName.lastIndexOf('.')+1,methName.indexOf('('));
    System.out.println("b2= "+methName);
Method meth = c.getMethod(methName, partypes);
System.out.println("a= "+methName);
    //invoke the method at runtime.
Object retobj= meth.invoke(cobj,par);
    //get the output of the method
Integer retval = (Integer)retobj;
    System.out.println(prin+" result^ "+retval);
}catch(Exception e){
    System.out.println(e);
}
}

//end of main
//}////end of class

The following method is called to involve the ComaprativeCorrectness object
Sa1 is the path of the class file and Sa2 is the name of the class file
public String compCorrectness(File f)
{
    System.out.println("inside comp= "+f);
    StringBuffer sb = new StringBuffer();
    try{
        String sa1 = f.getParent();
        System.out.println("sa1= "+sa1);
        String sa2 = f.getName();
        System.out.println("sa2= "+sa2);
        Runtime rt = Runtime.getRuntime();
        String sa3 = home1+"\bin\java -classpath "+sa1+" -D "+sa2;
        System.out.println("sa1= "+sa3);
        Process pp = rt.exec(sa3);
        System.out.println("pp= "+pp);
        //pp.waitFor();
        BufferedReader br = new BufferedReader(pp.getInputStream());
        int aa=0;
        System.out.println("br= "+br);
        while( (aa = br.read()) != -1)
sb.append((char)aa);

rt.gc();
rt.runFinalization();
}catch(Exception ee) { System.out.println("Compile Error :"+ee); return "Some problem occured while compiling. Please try again."; }
if(sb.length() == 0)
sb.append("Execution failed");
return sb.toString();
}
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import javax.swing.border.*;
import java.io.*;
import java.awt.geom.*;
import java.awt.image.*;
import javax.imageio.*;
import javax.swing.event.*;
public class ManGrading extends JFrame implements ActionListener
{
    static ManGrading pl;
    ScrollC sec;
    Font display = new Font("Serif", Font.BOLD, 12);
    final JTextArea TA = new JTextArea(6,50);
    JPanel p = new JPanel();
    JPanel contr = new JPanel();
    JPanel comment = new JPanel();
    class ListModelExample extends JPanel {
        JList listS;
        DefaultListModel modelS;
        JList listA;
        DefaultListModel modelA;
        JTextField valueFieldS;
        JTextField valueFieldA;
        JList liste;
        DefaultListModel modele;
        JTextField valueFielde;
        class ValueReporter implements ListSelectionListener {
            public void valueChanged(ListSelectionEvent event) {
                if (!event.getValueIsAdjusting())
                    valueFieldS.setText(listS.getSelectedValue().toString());
            }
        }
        class ValueReporterA implements ListSelectionListener {
            public void valueChanged(ListSelectionEvent event) {
                if (!event.getValueIsAdjusting())
                    valueFieldA.setText(listA.getSelectedValue().toString());
            }
        }
        class ValueReportere implements ListSelectionListener {
            public void valueChanged(ListSelectionEvent event) {
            }
        }
    }
}
if (!event.getValueIsAdjusting()) {
    valueField.setText(liste.getSelectedValue().toString());
    TA.setText("button 1");
}
}

public ListModelExample() {

setLayout(new BorderLayout());
Panel mainPanel = new Panel();
mainPanel.setLayout(new BorderLayout());
Panel sPanel = new Panel();
sPanel.setLayout(new BorderLayout());
models = new DefaultListModel();
listS = new JList(modelS);
HstPanelO.setBackground(Color.white);
Border HstPanelBorderO =
        BorderFactory.createTitledBorder("Student List");
listPanelO.setBorder(HstPanelBorderO);
mainPanel.add("Center", listPanelO);
Panel aPanel = new Panel();
aPanel.setLayout(new BorderLayout());
modelA = new DefaultListModel();
listA = new JList(modelA);
listPanelA = new JPanel();
listPanelA.setBackground(Color.white);
Border listPanelBorderA =
        BorderFactory.createTitledBorder("Assignment List");
listPanelA.setBorder(listPanelBorderA);
listPanelA.add(aPanel);
mainPanel.add("North", listPanelA);
Panel ePanel = new Panel();
ePanel.setLayout(new BorderLayout());
modele = new DefaultListModel();
liste = new JList(modele);
JPanel listPanel0e = new JPanel();
listPanel0e.setBackground(Color.white);
Border listPanelBorder0e =
        BorderFactory.createTitledBorder("Error Statistics");
listPanel0e.setBorder(listPanelBorder0e);
mainPanel.add("South", listPanel0e);
listS.setFont(display);
listS.setVisibleRowCount(4); // 3 enough?
listS.addListSelectionListener(new ValueReporter());
JScrollPane paneS = new JScrollPane(listS);

listA.setFont(display);
listA.setVisibleRowCount(3);
listA.addListSelectionListener(new ValueReporterA());
JScrollPane paneA = new JScrollPane(listA);

liste.setFont(display);
liste.setVisibleRowCount(4);
liste.addListSelectionListener(new ValueReportere());
JScrollPane panee = new JScrollPane(liste);

sPanel.add("North", paneS);
nextButtonS = new JButton("SHOW");
prevButtonS = new JButton("ORIGINIAL");
sPanel.add("Center", nextButtonS);
nextButtonA = new JButton("SHOW STUDENTS");
prevButtonA = new JButton("Xtra");
eaPanel.add("North", paneA);
nextButtonA = new JButton("Add");
prevButtonA = new JButton(" ");
eaPanel.add("East", nextButtonA);
eaPanel.add("West", prevButtonA);

displayF = new Font("Serif", Font.BOLD, 12);
valueLabel = new JLabel("ASSIGNMENT");
valueLabel.setFont(displayF);
valueFieldA = new JTextField("None", 17);
valuePanel = new JPanel();
valuePanel.setBackground(Color.white);
valuePanelBorder = BorderFactory.createTitledBorder("Selection");
valuePanel.setBorder(valuePanelBorder);
valuePanel.add(valueLabel);
valuePanel.add(valueFieldA);
aPanel.add("South", valuePanel);
setVisible(true);

valueLabelS = new JLabel("STUDENT");
valueLabelS.setFont(displayF);
valueFieldS = new JTextField("None", 17);
valueFieldS.setFont(displayF);
valuePanelS = new JPanel();
valuePanelS.setBackground(Color.white);
valuePanelSBorder = BorderFactory.createTitledBorder("Selection");

valuePanelS.setBorder(valuePanelBorderS);
valuePanelS.add(valueLabelS);
valuePanelS.add(valueFieldS);
sPanel.add("South", valuePanelS);
setVisible(true);
JLabel valueLabel = new JLabel("ERROR");
valueLabel.setFont(displayF);
valueField = new JTextField("None", 17);
valueField.setFont(displayF);
JPanel valuePanele = new JPanel();
valuePanele.setBackground(Color.white);
Border valuePanelBorder =
BorderFactory.createTitledBorder("Selection");
valuePanele.setBorder(valuePanelBorder);
valuePanele.add(valueLabel);
valuePanele.add(valueField);
ePanel.add("South", valuePanele);
setVisible(true);

// sampleJList2.setVisibleRowCount(4); // 4 should be enough otherwise may look complicated

// CONNECTION CONNECTION(1)
// get list of the students
// if there are more students automatically a scroll bar will come up

collections.addElement("Chris Donald ");
collections.addElement("Matt Alom ");
collections.addElement("Kathy ");
collections.addElement("Christina ");
collections.addElement("Assignment 1 ");
collections.addElement("Assignment 2 ");
collections.addElement("Assignment 3 ");
collections.addElement("Assignment 4 ");
collections.addElement("Error 1 ");
collections.addElement("Error 2 ");
collections.addElement("Error 3 ");
collections.addElement("Error 4 ");

// CONNECTION CONNECTION(2)

// at the mean time only adds a student

nextButtonS.addActionListener(
new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // add DB connections and get if this student's, this assignment is graded or not
// accordingly change the flagdisplay.
// 1=no pic so display from the file.
// 2=display the pic.
// 3=no pic.no file..at the time of loading.
System.out.println("pressed SHOW");
    scc.sc.c.flagdisplay=1;
    scc.sc.c.initpaint(scc.sc.c.buffImage.getGraphics());
System.out.println("pressed SHOW-after");
    scc.sc.c.repaint();
});

// CONNECTION CONNECTION(3)
// at the mean time only removes a stuedent
prevButtonS.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            if (models.getSize() > 0)
                models.removeElementAt(0);
        }
    });
nextButtonA.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            if (modelA.getSize() > 0)
                modelA.addElement("assignment ");
        }
    });

// CONNECTION CONNECTION(3)
// at the mean time only removes a stuedent
prevButtonA.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            if (modelA.getSize() > 0)
                modelA.removeElementAt(0);
        }
    });
nextButtonA.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            modele.addElement("errornew ");
        }
    });
CONNECTION CONNECTION(3)
    // at the mean time only removes a student
prevButtone.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            if (modele.getSize() > 0)
                modele.removeElementAt(0);
        }
    });

    //
    add(mainPanel, BorderLayout.NORTH);
}

public static PButton whatt;
public static JLabel whatc;
ListModelExample LMD = new ListModelExample();

public ManGrading()
{
    addWindowListener(new WindowAdapter()
    {
        public void windowClosing(WindowEvent ev)
        {
            dispose();
            System.exit(0);
        }
    });
    this.setTitle("Intelligent Online Assignment Evaluation");
    setBounds(0,0,1025,740);

    JPanel left = new JPanel();
left.setBackground(Color.white);
left.add(LMD);
//left.add(panel);
sec = new ScrollC();
//myc = sec.sc.c;
getContentPane().add("West",left);
getContentPane().add("East",contr);

    // Scrollbar sv = new Scrollbar
    // (Scrollbar.VERTICAL,0, 1, 100,1000); //rh);
// panel.add("West", sv);

// getContentPane().add("Center",panel);

getContentPane().add("Center",scc);

//final JTextArea TA = new JTextArea(6,50);
JScrollPane SP = new JScrollPane(TA);
comment.setBackground(Color.white);
comment.add("Center",SP);

final JTextArea TAG = new JTextArea(6,6);
JScrollPane SPG = new JScrollPane(TAG);
//TAG.setText("Grade");
JButton TABG = new JButton("Save Grade");
JPanel gpane = new JPanel();
gpane.setBackground(Color.white);
Border g = BorderFactory.createTitledBorder("Grade:");
gpane.setBorder(g);

gpane.add("Center",SPG);
gpane.add("East",TABG);

TABG.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            TAG.setText("Grade");
        }
    });

JPanel com = new JPanel();
com.setBackground(Color.white);
com.add("Center",comment);
com.add("East", gpane);

Border b = BorderFactory.createTitledBorder("Comments:");
comment.setBorder(b);

getContentPane().add("South",com);

JButton TAB1 = new JButton("Save Comment");
JButton TAB2 = new JButton("Create new Error type");
JPanel buttonP = new JPanel();
buttonP.setBackground(Color.white);
buttonP.setLayout(new GridLayout(2,1));
buttonP.add(TAB1);
buttonP.add(TAB2);

TAB1.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            TA.setText("button 1");
        }
    });

TAB2.addActionListener(
    new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            TA.setText("button 2");
        }
    });

comment.add("East", buttonP);

//comment.add("West", TAB2);

contr.setLayout(new BorderLayout());
Border br = BorderFactory.createEtchedBorder(EtchedBorder.LOWERED, Color.white, Color.gray);
contr.setBorder(br);
contr.add("North", addButtons());
contr.add("Center", new JPanel());
contr.add("South", addColors());
setVisible(true);
panel.requestFocusO;
}
private JPanel addButtons()
{
    final JPanel panelb = new JPanel();
    panelb.setPrefereedSize(new Dimension(70, 175));
    panelb.setLayout(new GridLayout(5, 2, l, l));
    for (int i = 0; i < 3; i++)
        {
        PButton jb = new PButton(i);
        jb.addActionListener(this);
        panelb.add(jb);
        
    }
JButton btnsave = new JButton("SAVE");
JButton btnundo = new JButton("UNDO");
//btnsave.addActionListener(this);

//action listener for UNDO
btnundo.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            System.out.println("undo clicked obj=
"+panel.objects.lastIndexOf(panel.objects.lastElement()));

            //panel.objects.remove(panel.objects.lastIndexOf(panel.objects.lastElement()));
            //panel.repaint();
            // Graphics g = panel.getGraphics();
            // panel.paint(g);
            }catch(Exception exce){}
    }
});
panelb.add(btnundo);

//action listener for save
btnsave.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            FileDialog fd = new FileDialog(pl, "Save as PNG", FileDialog.SAVE);
            fd.setFile("canvas.png");
            fd.show();
            String name = fd.getFile();
            Rectangle r = scc.sc.c.getBounds();
            System.out.println("w= "r.width+" h= "r.height+" spaceBLines+sc.c.spaceBLines);"+sc.c.spaceBLines);
            //Image image = sc.c.createImage(r.width, sc.c.spaceBLines);
            Graphics g = sc.c.buffImage.getGraphics();
            // Graphics g = sc.c.offscreen;
            //sc.c.paint(g);
            //System.out.println("name ="+name);
            ImageIO.write((RenderedImage)sc.c.buffImage, "png", new
File(name));
        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
    }
});
```java
Panelb.add(btnSave);
return(Panelb);

private JPanel addColors()
{
    JPanel panel = new JPanel();
    panel.setLayout(new BorderLayout());
    JPanel cancel = new JPanel();
    panel.add("North",cancel);
    cancel.setLayout(new GridLayout(5,4,0,0));
    cancel.setPreferredSize(new Dimension(70,100));
    Color[] colors = {Color.black, Color.blue,
                     new Color(152,0,0),new Color(132,66,0),
                     Color.cyan, new Color(219,219,112),
                     Color.gray,Color.green,
                     new Color(200,200,200),
                     Color.magenta,new Color(0,0,108),
                     Color.orange,Color.pink,
                     Color.red,new Color(155,192,210),
                     new Color(230,230,230),new Color(64,224,208),
                     Color.white,Color.yellow};
    for (int i=0; i < colors.length; i++)
    {
        JButton jb = new JButton(" ");
        jb.setBackground(colors[i]);
        jb.addActionListener(this);
        cancel.add(jb);
    }
    Whatc = new JLabel(" ");
    Whatc.setOpaque(true);
    Whatc.setBackground(Color.blue);
    Whatc.setBorder(new MatteBorder(1,1,1,1,Color.black));
    panel.add("Center",whatc);
    return(panel);
// end of add color
}
public void actionPerformed(ActionEvent ae)
{
    if (ae.getSource() instanceof PButton)
    {
```
whatt = (PButton) ae.getSource();
    // System.out.println("type= "+whatt.type);

// System.out.println("+whatt.type);

public class PButton extends JButton
{
    int type;

    public PButton(int type)
    {
        this.type = type;
    }

    public PButton(String name)
    {
        super(name);
    }

    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);
        // System.out.println("I am in 2nd pc");
        Graphics2D g2 = (Graphics2D)g;
        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
                            RenderingHints.VALUE_ANTIALIAS_ON);
        g2.setStroke(new BasicStroke(1.2f));
        if (type == 0) g.drawLine(8,8,getWidth()-10,getHeight()-11);
        if (type == 1) g.drawOval(6,10,getWidth()-13,getHeight()-20);
        if (type == 2) g.drawRect(7,7,getWidth()-15,getHeight()-14);
        if (type == 3) g.drawString("UNDO",getWidth()-55,getHeight()-12);
        if (hasFocus())
        {
            g.setColor(Color.gray);
            g.drawRect(4,3,getWidth()-9,getHeight()-7);
        }
    }

    public class PObject extends JComponent
{  
    int   type;  
    Point fp,tp;  
    Color  color;  
    boolean fill;

    public PObject(int type, Point fp, Point tp, Color color)
    {
        this.type = type;
        this.fp    = fp;
        this.tp    = tp;
        this.color = color;
    }

    public void draw(Graphics g)
    {
        g.setColor(color);
        if (type == 0) g.drawLine(fp.x,fp.y,tp.x,tp.y);
        if (type == 1) g.drawOval(fp.x,fp.y,tp.x-fp.x,tp.y-fp.y);
        if (type == 2) g.drawRect(fp.x,fp.y,tp.x-fp.x,tp.y-fp.y);
    }
}

public static void main (String[] args)
{
    pl= new ManGrading();
}

}

import java.applet.*;
import java.awt.*;
/*
class ScrollC extends JPanel //java.applet.Applet
{
    ScrollCanvas sc;

    // public void init() {
}
public ScrollC() {
    setLayout(new FlowLayout());
    sc = new ScrollCanvas
        (600,550, 750,750, Color.white,
        Color.black);//Width,550,750,height
    add(sc);
}

public boolean handleEvent(Event e) {
    if (e.target instanceof Scrollbar) {
        switch (e.id) {
            case Event.SCROLL_ABSOLUTE: 
            case Event.SCROLL_PAGE_DOWN: 
            case Event.SCROLL_PAGE_UP: 
            case Event.SCROLL_LINE_UP:
            case Event.SCROLL_LINE_DOWN: 
                sc.redraw(); 
                return true;
        }
    }
    return super.handleEvent(e);
}

class ScrollCanvas extends Panel {
    int vw,vh;
    int rw,rh;
    Color b,f;
    myCanvas c;
    Scrollbar sv, sh;

    // constructor
    // visible h w
    // real  h w
    // background foreground
    ScrollCanvas 
        (int vw1, int vh1, int rw1, int rh1, Color b1, Color f1) {
        super();
        vw = vw1; vh = vh1;
        rh = rh1; rw = rw1;
        b = b1; f = f1;
        int ScrollIncrement = 10;
        setLayout(new BorderLayout());
        c = new myCanvas(vw, vh, rw, rh, b ,f);

        sv = new Scrollbar
(Scrollbar.VERTICAL, 0, 1, 0, rh);

add("East", sv);
add("West", c);

sh = new Scrollbar
(Scrollbar.HORIZONTAL, 0, ScrollIncrement, 0, rw);
add("South", sh);

public void redraw() {
  int y = sv.getValue();
  int x = sh.getValue();
  c.setNewPoint(x, y);
  c.draw(x, y);
}

public Dimension minimumSize() {
  return new Dimension(vw, vh);
}

public Dimension preferredSize() {
  return new Dimension(vw, vh);
}

}

class myCanvas extends Canvas implements MouseMotionListener, MouseListener{
  Point mp, fp, tp;
  Vector objects = new Vector();

  int spaceBLines;
  int vw, vh;
  int rw, rh;
  Color b, f;
  int x, y;
  int xN, yN;
  static int h;
  Image buffImage;
  public static Graphics offscreen;
  int flagdisplay = 3;
  // 1 = no pic so display from the file.
  // 2 = display the pic.
  // 3 = no pic no file..at the time of loading.
  myCanvas
      (int vw1, int vh1, int rw1, int rh1, Color b1, Color f1) {

super();
addMouseMotionListener(this);
    addMouseListener(this);
vw = vwl; vh = vh1;
rh = rhl; rw = rwl;
//System.out.println("vw= " +vw+" vh= "+vh+" rh= "+rh+" rw= "+rw);
b = b1; f = fl1;
// initDone = false;
repaint();
} // constructor

public void setNewPoint(int x, int y){
    h = y;
}
public void paint(Graphics g) {
    if (flagdisplay==1){
        System.out.println("i am painting again");
        initpaint(g);
        System.out.println("iam in 1st pc");
        System.out.println("obj-size= "+objects.size());
        for (int i=0; i < objects.size(); i++)
        {
            ManGrading.PObject po = (ManGrading.PObject)objects.get(i);
            po.draw(offscreen);
        }
    g.setColor(ManGrading.whatc.getBackground());
    if (ManGrading.whatt != null && fp != null) {
        System.out.println("iam inside not null");
        if (ManGrading.whatt.type == 0) g.drawLine(fp.x,fp.y,tp.x,tp.y);
        if (ManGrading.whatt.type == 1) g.drawOval(fp.x,fp.y,tp.x-fp.x,tp.y-fp.y);
        if (ManGrading.whatt.type == 2) offscreen.drawRect(fp.x,fp.y,tp.x-fp.x,tp.y-fp.y);
    }
    // }
    //end of if
    //end of paint()

public void update(Graphics g) {
    if (flagdisplay==1) {
        System.out.println("flag is 1 in update");
        g.drawImage(buffImage, x, y, this);
        // }
    }
}

public void initpaint(Graphics g) {
//1=no pic so display from the file.
//2=display the pic.
//3=no pic.no file..at the time of loading.
String fileread="C:/first.txt";
if (flagdisplay==1) {
    fileread="C:/readme.txt";
}
System.out.println("flag is 1");
String info1 = "";

Vector vector = new Vector();

try {
    FileReader fr = new FileReader(fileread);
    BufferedReader inFile = new BufferedReader(fr);
    String line = inFile.readLine();
    while(line != null) {
        info1+=line+"\n";
        vector.addElement(line);
        line = inFile.readLine();
    }
}
catch(Exception eio) {System.out.println("ioexception "+eio);}

spaceBLines=20;

try {
    buffImage = this.createImage(rw, rh);
    System.out.println("before");
    offscreen = buffImage.getGraphics();
    offscreen.setColor(b);
    offscreen.fillRect(0, 0, rw-10, rh-10);
    offscreen.setColor(f);
//System.out.println("after");
offscreen.setFont(new Font("Courier", Font.BOLD, 12));
//System.out.println("after2");
for(int i=0;i<vector.size();i++){
//System.out.println("after3");
offscreen.drawString(i+" "+vector.elementAt(i),10, spaceBLines);
spaceBLines+=spaceBLines+20;
}
System.out.println("buffImage = "+buffImage);
System.out.println("this = "+this);
System.out.println("g = "+g);
g.drawImage(buffImage,0,0, this);
System.out.println("this-after = ");
}
catch (Exception e) {
        System.out.println("oups..."+e);
}
//} //end of if flagdisplay
//end of initpaint

public void draw (int x1, int y1) {
    x = -x1;
y = -y1;
update(getGraphics());
}

public Dimension minimumSize() {
    return new Dimension(vw,vh);
}

public Dimension preferredSize() {
    return new Dimension(vw,vh);
}

public void mouseDragged(MouseEvent m) {
    System.out.println("its me..being dragged");
    if (ManGrading.whatt == null){ System.out.println("its return");return;}
    if (fp == null)
    {System.out.println("fp is null-here");
        fp = new Point();
        tp = new Point();
    }
    if (ManGrading.whatt.type == 0)
    {fp.setLocation(mp.getLocation());
    }
tp.setLocation(m.getPoint().getLocation());
}
else
{System.out.println("its nopee");
    fp.x = Math.min(mp.x,m.getX());
    fp.y = Math.min(mp.y,m.getY())+h;
    tp.x = Math.max(mp.x,m.getX());
    tp.y = Math.max(mp.y,m.getY())+h;
}
}

public void mouseMoved(MouseEvent m){}
public void mouseClicked(MouseEvent m){}
public void mouseEntered(MouseEvent m){}
public void mouseExited(MouseEvent m) {}
public void mouseReleased(MouseEvent m)
{
    if (ManGrading.whatt == null) return;
    if (fp != null){
        System.out.println("released");
        int whatttype= ManGrading.whatt.type;
        Color whatcback= ManGrading.whatc.getBackground();
        objects.add( ManGrading.pl.new PObject(whatttype,fp,tp,whatcback));
        paint(offscreen);
        fp = null;
    }
}
public void mousePressed(MouseEvent m)
{
    System.out.println("pressed");
    mp = m.getPoint(); //mp.y=mp.y+h;
    paint(offscreen);
}

}//end of mycanvas class
15. References


[16] www.woodger.ca


[20] Tim Reeves, Paula Baxter, Cheryl Jordan (October 2002) "Teaching computing courses - computer literacy, business microcomputer applications, and introduction to programming online utilizing webCT" Journal of Computing Sciences in Colleges, Volume 18 Issue 1


