Krishnaraj OOPCJ Assignment 1

Krishnaraj OOPCJ Assignment 2

Krishnaraj OOPCJ Assignment 3

Krishnaraj OOPCJ Assignment 4

Krishnaraj OOPCJ Assignment 5

Krishnaraj OOPCJ Assignment 6

Krishnaraj OOPCJ Assignment 7

Krishnaraj OOPCJ Assignment 8

Krishnaraj OOPCJ Assignment 9

Krishnaraj OOPCJ Theory Assignment 1

Krishnaraj OOPCJ Theory Assignment 2

Krishnaraj Case Study Report

Krishnaraj Project Report

MIT World Peace University

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 3

To demonstrate the use of objects, classes, constructors and destructors using C++ and JAVA.

Practical Report

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

November 29, 2022

Contents

1 Aim and Objectives

To demonstrate the use of objects, classes, constructors and destructors using C++ and JAVA.

1. To study various OOP concepts
2. To acquaint with the use of objects and classes in C++ and Java.
3. To learn implementation of constructor, destructors and dynamic memory allocation

2 Problem Statement

Develop an object-oriented program to create a database of employee information system containing the following information: Employee Name, Employee number, qualification, address, contact number, salary details (basic, DA, TA, Net salary), etc. Construct the database with suitable member functions for initializing and destroying the data viz. constructor, default constructor, Copy constructor, destructor. Use dynamic memory allocation concept while creating and destroying the object of a class. Use static data member concept wherever required. Accept and display the information of Employees.

3 Theory

3.1 Algorithm

An *Algorithm* is a set of statements for solving a problem. They're the building blocks for programming, and they allow things like computers, smartphones, and websites to function and make decisions.

It looks like this, and can be generally implemented in any suitable programming language.

```
step 1: Start
step 2: Do something
step 3: Do something else
step 4: Return result
step 5: Stop
```

3.2 Class

1. In object-oriented programming, a *class* is a blueprint for creating objects (a particular data structure), providing initial values for state (member variables or attributes), and implementations of behavior (member functions or methods).
2. It is a basic concept of Object-Oriented Programming which revolve around the real-life entities.
3. It is like a template for creating objects. You can initialize various variables of varying data types in it.
4. The Syntax for C++ and Java is given below.

```cpp
// Syntax in C++
class Name_of_Class
{
  private:
    int a;
  public:
    int b;
  protected:
    int c;
    data_type member_function()
    {
      return data_type;
    }
}objects;
```

```cpp
// Syntax in Java
class Name_of_Class
{
  private:
    int a;
  public:
    int b;
  protected:
    int c;
    data_type member_function()
    {
      return data_type;
    }
}
```

3.3 Objects

1. *Object* is an instance of a *class*. An object in *OOP* is nothing but a self-contained component which consists of methods and properties to make a particular type of data useful.

2. For example color name, table, bag, barking. When you send a message to an object, you are asking the object to invoke or execute one of its methods as defined in the class.

3. From a programming point of view, an object in OOPS can include a data structure, a variable, or a function. It has a memory location allocated. Java Objects are designed as class hierarchies.

4. The syntax for Java and C++ are given below.

```cpp
// C++ Syntax
class_name obj;
Employee CEO;
Employee President(Name, Age);
```

```cpp
// Java Syntax
class_name obj = new class_name;
Employee CEO = new Employee();
Employee President = new Employee(Name, Age);
```

3.4 Default, Parameterized, and Copy Construtor

A constructor is a special member function with exact same name as the class name.
There are **3** Types of Constructors:

1. **Default Constructor**: A constructor with no arguments (or parameters) in the definition is a default constructor. Usually, these constructors use to initialize data members (variables) with real values. If no constructor is explicitly declared, the compiler automatically creates a default constructor with no data member (variables) or initialization.
2. **Parameterized Constructor**: Unlike Default constructor, It contains parameters (or arguments) in the constructor definition and declaration. More than one argument can also pass through a parameterized constructor. Moreover, these types of constructors use for overloading to differentiate between multiple constructors.
3. **Copy Constructor**: A copy constructor is a member function that initializes an object using another object of the same class. It helps to copy data from one object to another.
4. The Syntax for Each of them is same for C++ and Java and is given below.

```
class Avenger
{
  static int hero = 0;
  int age;
  bool from_earth, is_a_God;
  string Superpower;

  // default Constructor
  Avenger()
  {
    hero++;
  }

  // Parameterized Constructor
  Avenger(bool from_earth, bool is_a_God, int age, String name, string
  Superpower)
  {
    if(!from_earth)
    {
      alien_avengers++;
    }
  }

  // Copy Constructor

  Avenger(Avenger &new_Avenger)
  {
    this.superpower = new_Avenger.superpower;
  }
}
```

3.5 Destuctors

Destructor is an instance member function which is invoked automatically whenever an object is going to be destroyed. Meaning, a destructor is the last function that is going to be called before an object is destroyed.

1. Destructor is also a *special member* function like constructor. Destructor destroys the class objects created by constructor.
2. Destructor has the same name as their class name preceded by a tiled ( ) symbol.
3. It is not possible to define more than one destructor.
4. The destructor is only one way to destroy the object create by constructor. Hence destructor can-not be overloaded.
5. Destructor neither requires any argument nor returns any value.
6. It is automatically called when object goes out of scope.
7. Destructor release memory space occupied by the objects created by constructor.
8. In destructor, objects are destroyed in the reverse of an object creation.

In Java, The job of the constructor is done by the compiler, as garbage collection in general is done better by the compiler in java than the programmer. It is for this reason that the finalize() function exists in java but is depricated and its strongly recommended not to use it.

```
// C++ Implementation
class Avenger()
{
  ~Avenger()
  {
    pay_respects();
  }

  // in java
  finalize()
  {
    System.gc() // call the garbage collector
  }
}
```

3.6 Dynamic Allocation and DeAllocation

1. Many times, you are not aware in advance how much memory you will need to store particular information in a defined variable and the size of required memory can be determined at run time.
2. You can allocate memory at run time within the heap for the variable of a given type using a special operator in C++ which returns the address of the space allocated. This operator is called new operator.
3. If you are not in need of dynamically allocated memory anymore, you can use delete operator, which de-allocates memory that was previously allocated by new operator.
4. The Syntax for C++ and Java is given below.

```
1
2  // C++
3  int a = new int;
4  delete a;
5
6  // Java
7  Employee Obj = new Employee()
```

4 Algorithm

1. **Start**
2. Create Employee Class
3. Delcare appropirate Data Memebers and define the member funtions
4. Accept multiple employee's Data using an Array of Objects
5. Assign some Basic employees using default, copy and parameterized constructors
6. Show usage of Default Constructor and display the Data
7. Show usage of Parameterized constructor and display that data
8. Show usage of Copy Constructor and display that Data.
9. Distory the objects if possible
10. **End**

5 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers : g++ and gcc on linux for C++, and javac, with JDK 18.0.2 for Java

6 Input

1. Number of Employees for enrollment
2. Employee ID
3. Employee Name
4. Employee Position
5. Employee Address
6. Employee Salary

7 Output

Employee Data should be displayed by use of member functions.

8 Conclusion

Thus, learned to use objects, classes, constructor and destructor and implemented solution of the given problem statement using C++ and Java.

9 Code

9.1 Java Implementation

```java
// Assignment 1 - OOPCJ
// Write A Program to show implementation of Constructors in Java
// PA34. Krishnaraj Thadesar
// SY CSE Semester 3
// Date Of Implementation: 19th August


package assignment_1;
import java.util.Scanner;

class Employee {

    // create an object of Scanner
    Scanner input = new Scanner(System.in);

    int emp_id;
    int age, basic_sal, da, ta;
    String address_city, position, name;
    static int ssn;

    Employee()
    {
        System.out.println("Default Constructor was called");
    }

    // Parameterized Constructor
    Employee(int e, int a, int b, int d,
    int t, String add, String pos, String nam)
    {
        System.out.println("Parameterized constructor was called");
        emp_id = e;
        age = a;
        basic_sal = b;
        da = d;
        ta = t;
        address_city = add;
        position = pos;
        name = nam;
    }

    // Copy Constructor
    Employee(Employee E)
```

```
43      {
44          System.out.println("Copy constructor was called");
45          emp_id = E.emp_id;
46          age = E.age;
47          basic_sal = E.basic_sal;
48          da = E.da;
49          ta = E.ta;
50          address_city = E.address_city;
51          position = E.position;
52          name = E.name;
53      }
54
55      double calc_gross_sal()
56      {
57          return basic_sal + da + ta - (0.15 * basic_sal);
58      }
59
60      void display()
61      {
62          ssn = ssn + 1;
63          System.out.println("Employee ssn is: " + ssn);
64          System.out.println("Employee ID is : " + emp_id);
65          System.out.println("Employee Name: " + name);
66          System.out.println("Employee Age: " + age);
67          System.out.println("Employee Position: " + position);
68          System.out.println("Employee basic Salary: " + basic_sal);
69          System.out.println("Employee DA: " + da);
70          System.out.println("Employee TA: " + ta);
71          System.out.println("Employee Gross Salary: " + calc_gross_sal() );
72          System.out.println("Employee Address City: " + address_city);
73          System.out.println("\n");
74      }
75
76      void accept()
77      {
78          System.out.println("Enter the age :");
79          age = input.nextInt();
80          System.out.println("Employee ID is: ");
81          emp_id = input.nextInt();
82          System.out.println("Employee Name: " );
83          name = input.next();
84          System.out.println("Employee Age: " );
85          age = input.nextInt();
86          System.out.println("Employee Position: " );
87          position = input.next();
88          System.out.println("Employee basic Salary: ");
89          basic_sal = input.nextInt();
90          System.out.println("Employee DA: ");
91          da = input.nextInt();
92          System.out.println("Employee TA: ");
93          ta = input.nextInt();
94          System.out.println("Employee Address City: ");
95          address_city = input.next();
96      }
97
98      // @Override
99      // protected void finalize() throws Throwable
100     // {
101     //     try
```

```
    //      {
    //          input.close();
    //      }
    //      catch(Throwable t)
    //      {
    //          throw t;
    //      }
    //      finally
    //      {
    //          super.finalize();
    //      }
    // }

}
```

Listing 1: Employee.java

```
// Develop an object-oriented program to create a database of employee information system
// containing the following information: Employee Name, Employee number, qualification,
// address, contact number, salary details (basic, DA, TA, Net salary), etc. Construct the
// database with suitable member functions for initializing and destroying the data viz.
// constructor, default constructor, Copy constructor, destructor. Use dynamic memory
// allocation concept while creating and destroying the object of a class. Use static data
// member concept wherever required. Accept and display the information of Employees.

package assignment_1;

import java.util.Scanner;

public class Source {

    static Scanner input = new Scanner(System.in);

    public static void main(String[] args) {
        System.out.println("This is the first Assignment");
        int count = 0;

        // Default Constructor
        Employee CEO = new Employee();
        CEO.emp_id = 1000;
        CEO.name = "Kom Pany Seeio";
        CEO.address_city = "Seoul";
        CEO.age = 45;
        CEO.basic_sal = 1000000;
        CEO.da = 1000;
        CEO.ta = 2000;
        CEO.position = "CEO";

        // Defining an object using the copy constructor
        Employee President = new Employee(CEO);
        President.name = "Precy Dent";
        President.age = 45;
```

```
        President.address_city = "Delhi";
        President.basic_sal *= 2;
        President.position = "President";

        Employee VP = new Employee(1003, 50, 200000, 3000, 1000,
                "Mumbai", "Vice President", "Visey Presed Ent");

        System.out.println("Information about the CEO");
        CEO.display();
        System.out.println("Information about the President");
        President.display();
        System.out.println("Information about the President");
        VP.display();

        System.out.println("Enter the number of employees :");
        count = input.nextInt();
        Employee objs[] = new Employee[count];

        for (int i = 0; i < count; i++) {
            objs[i] = new Employee();
            objs[i].accept();
            objs[i].display();
        }
        System.gc();
    }
}
```

Listing 2: Source.java

9.1.1 Java Input

```
Enter the number of employees :
2
Default Constructor was called
Enter the age :
35
Employee ID is:
006
Employee Name:
Peter
Employee Age:
17
Employee Position:
Avenger
Employee basic Salary:
500000
Employee DA:
3440
Employee TA:
3550
Employee Address City:
Brooklyn


Default Constructor was called
Enter the age :
4
Employee ID is:
007
```

```
Employee Name:
Thor
Employee Age:
1500
Employee Position:
Avenger
Employee basic Salary:
6000000
Employee DA:
3000
Employee TA:
5000
Employee Address City:
Asgard
```

Listing 3: Python example

9.1.2 Java Output

```
This is the first Assignment
Default Constructor was called
Copy constructor was called
Parameterized constructor was called
Information about the CEO
Employee ssn is: 1
Employee ID is : 1000
Employee Name: Kom Pany Seeio
Employee Age: 45
Employee Position: CEO
Employee basic Salary: 1000000
Employee DA: 1000
Employee TA: 2000
Employee Gross Salary: 853000.0
Employee Address City: Seoul


Information about the President
Employee ssn is: 2
Employee ID is : 1000
Employee Name: Precy Dent
Employee Age: 45
Employee Position: President
Employee basic Salary: 2000000
Employee DA: 1000
Employee TA: 2000
Employee Gross Salary: 1703000.0
Employee Address City: Delhi


Information about the President
Employee ssn is: 3
Employee ID is : 1003
Employee Name: Visey Presed Ent
Employee Age: 50
Employee Position: Vice President
Employee basic Salary: 200000
Employee DA: 3000
Employee TA: 1000
```

```
Employee Gross Salary: 174000.0
Employee Address City: Mumbai


Employee ssn is: 4
Employee ID is : 6
Employee Name: Peter
Employee Age: 17
Employee Position: Avenger
Employee basic Salary: 500000
Employee DA: 3440
Employee TA: 3550
Employee Gross Salary: 431990.0
Employee Address City: Brooklyn


Employee ssn is: 5
Employee ID is : 7
Employee Name: Thor
Employee Age: 1500
Employee Position: Avenger
Employee basic Salary: 6000000
Employee DA: 3000
Employee TA: 5000
Employee Gross Salary: 5108000.0
Employee Address City: Asgard
```

Listing 4: Python example

9.2 C++ Implementation

```
// Assignment 1 - OOPCJ
// Write A Program to show implementation of Constructors in C++
// PA34. Krishnaraj Thadesar
// SY CSE Semester 3
// Date Of Implementation: 17th August

// Develop an object-oriented program to create a database of employee information system
// containing the following information: Employee Name, Employee number, qualification,
// address, contact number, salary details (basic, DA, TA, Net salary), etc. Construct the
// database with suitable member functions for initializing and destroying the data viz.
// constructor, default constructor, Copy constructor, destructor. Use dynamic memory
// allocation concept while creating and destroying the object of a class. Use static data
// member concept wherever required Accept and display the information of Employees.

#include <iostream>
using namespace std;
class Employee
{
private:
    static int ssn;
```

```
21
22 protected:
23     int something = 5;
24
25 public:
26     int emp_id = 1000;
27     int age = 0, basic_sal = 0, da = 0, ta = 0;
28     string address_city, position, name;
29
30     // Default Constructor
31     Employee()
32     {
33         ssn += 1;
34         cout << "The Default Constructor was called" << endl;
35     }
36
37     // Parameterized Constructor
38     Employee(int e, int a, int b, int d, int t, string add, string pos, string nam
       )
39     {
40         cout << "Parameterized constructor was called\n";
41         emp_id = e;
42         age = a;
43         basic_sal = b;
44         da = d;
45         ta = t;
46         address_city = add;
47         position = pos;
48         name = nam;
49     }
50
51     // Copy Constructor
52     Employee(Employee &E)
53     {
54         cout << "Copy Constructor was called" << endl;
55         emp_id = E.emp_id;
56         age = E.age;
57         basic_sal = E.basic_sal;
58         da = E.da;
59         ta = E.ta;
60         address_city = E.address_city;
61         position = E.position;
62         name = E.name;
63     }
64
65     void display()
66     {
67         cout << "Employee ssn is:" << ssn << endl;
68         cout << "Employee ID is : " << emp_id << endl;
69         cout << "Employee Name: " << name << endl;
70         cout << "Employee Age: " << age << endl;
71         cout << "Employee Position: " << position << endl;
72         cout << "Employee basic Salary: " << basic_sal << endl;
73         cout << "Employee DA: " << da << endl;
74         cout << "Employee TA: " << ta << endl;
75         cout << "Employee Gross Salary: " << calc_gross_sal() << endl;
76         cout << "Employee Address City: " << address_city << endl;
77     }
78
```

```
79      void accept()
80      {
81          cout << "Enter the Employee ID: " << endl;
82          cin >> emp_id;
83          cout << "Enter the Employee Name: " << endl;
84          cin >> name;
85          cout << "Enter the Employee Age: " << endl;
86          cin >> age;
87          cout << "Enter the Employee Position: " << endl;
88          cin >> position;
89          cout << "Enter the Employee basic Salary: " << endl;
90          cin >> basic_sal;
91          cout << "Enter the Employee DA: " << endl;
92          cin >> da;
93          cout << "Enter the Employee TA: " << endl;
94          cin >> ta;
95          cout << "Enter the Employee Address City: " << endl;
96          cin >> address_city;
97      }
98
99      float calc_gross_sal()
100     {
101         int gross_sal = da + ta + basic_sal - (.15 * basic_sal);
102         return gross_sal;
103     }
104
105     // Destructor
106     ~Employee()
107     {
108         cout << "The Destructor was called" << endl;
109     }
110 };
111
112 int Employee::ssn = 1000;
113
114 int main()
115 {
116     // Defining an Object using the default Constructor
117     Employee CEO;
118     CEO.emp_id = 1000;
119     CEO.name = "Kom Pany Seeio";
120     CEO.address_city = "Seoul";
121     CEO.age = 45;
122     CEO.basic_sal = 1000000;
123     CEO.da = 1000;
124     CEO.ta = 2000;
125     CEO.position = "CEO";
126
127     // Defining an object using the copy constructor
128     Employee President(CEO);
129     President.name = "Precy Dent";
130     President.age = 45;
131     President.address_city = "Delhi";
132     President.basic_sal *= 2;
133     President.position = "President";
134
135     // Defining anothe object using the parameterized constructor
136     Employee VP(1003, 50, 200000, 3000, 1000, "Mumbai", "Vice President", "Visey
    Presed Ent");
```

```
137
138     // Taking user input for the number of employees
139     int count = 0;
140     cout << "How many values do you wanna input ? ";
141     cin >> count;
142
143     // Dynamically Creating new Objects
144     Employee *Obj = new Employee[count];
145     cout << "Enter the Details" << endl;
146
147     // Accepting Objects from the User.
148     for (int i = 0; i < count; i++)
149     {
150         cout << "Enter information about the Employee Number: " << i + 1 << endl;
151         Obj[i].accept();
152     }
153
154     // Displaying Information about the CEO and the President
155     cout << "Information about the CEO" << endl;
156     CEO.display();
157     cout << "Information about the President" << endl;
158     President.display();
159     cout << "Information about the Vice President" << endl;
160     VP.display();
161
162     // Displaying Information about the other Employees
163     for (int i = 0; i < count; i++)
164     {
165         cout << "Information about the Employee Number: " << i + 1 << endl;
166         Obj[i].display();
167     }
168
169     delete[] Obj;
170     return 0;
171 }
```

Listing 5: Main.Cpp

9.2.1 C++ Input

```
1  The Default Constructor was called
2  Copy Constructor was called
3  Parameterized constructor was called
4  How many values do you wanna input ? 2
5  The Default Constructor was called
6  The Default Constructor was called
7
8  Enter the Details
9
10 Enter information about the Employee Number: 1
11 Enter the Employee ID:
12 005
13 Enter the Employee Name:
14 Tony
15 Enter the Employee Age:
16 40
17 Enter the Employee Position:
18 Philanthropist
19 Enter the Employee basic Salary:
```

```
5000000
Enter the Employee DA:
3000
Enter the Employee TA:
3000
Enter the Employee Address City:
NewYork

Enter information about the Employee Number: 2
Enter the Employee ID:
006
Enter the Employee Name:
Steve
Enter the Employee Age:
105
Enter the Employee Position:
Captain
Enter the Employee basic Salary:
600000
Enter the Employee DA:
2999
Enter the Employee TA:
2000
Enter the Employee Address City:
Brooklyn
```

Listing 6: C++ Input

9.2.2 C++ Output

```
Information about the CEO
Employee ssn is:1003
Employee ID is : 1000
Employee Name: Kom Pany Seeio
Employee Age: 45
Employee Position: CEO
Employee basic Salary: 1000000
Employee DA: 1000
Employee TA: 2000
Employee Gross Salary: 853000
Employee Address City: Seoul

Information about the President
Employee ssn is:1003
Employee ID is : 1000
Employee Name: Precy Dent
Employee Age: 45
Employee Position: President
Employee basic Salary: 2000000
Employee DA: 1000
Employee TA: 2000
Employee Gross Salary: 1.703e+06
Employee Address City: Delhi

Information about the Vice President
Employee ssn is:1003
Employee ID is : 1003
Employee Name: Visey Presed Ent
```

```
Employee Age: 50
Employee Position: Vice President
Employee basic Salary: 200000
Employee DA: 3000
Employee TA: 1000
Employee Gross Salary: 174000
Employee Address City: Mumbai

Information about the Employee Number: 1
Employee ssn is:1003
Employee ID is : 5
Employee Name: Tony
Employee Age: 40
Employee Position: Philanthropist
Employee basic Salary: 5000000
Employee DA: 3000
Employee TA: 3000
Employee Gross Salary: 4.256e+06
Employee Address City: NewYork

Information about the Employee Number: 2
Employee ssn is:1003
Employee ID is : 6
Employee Name: Steve
Employee Age: 105
Employee Position: Captain
Employee basic Salary: 600000
Employee DA: 2999
Employee TA: 2000
Employee Gross Salary: 514999
Employee Address City: Brooklyn

The Destructor was called
The Destructor was called
The Destructor was called
The Destructor was called
The Destructor was called
```

Listing 7: C++ Output

10 FAQs

1. **What are classes?**

   In object-oriented programming, a **class** is a blueprint for creating objects (a particular data structure), providing initial values for state (member variables or attributes), and implementations of behavior (member functions or methods).
   It is a basic concept of Object-Oriented Programming which revolve around the real-life entities.

   ```
   class <class_name>{
   field;
   method;
      }
   ```

2. **Explain : Array of Objects:**

   An array of objects is like any other array in C++ and Java. An Array usually is just a collection of variables that have the same data type, and are placed in contiguous memory locations. An Array of Objects is similar in that instead of variables there are objects which are placed contigiously in memory.

   ```
   Syntax:
   Employee obj[5];
   ```

3. **Explain when to use different types of constructors? There are 3 Types of constructors**:

   (a) **Default Constructor**: This type is called when the Object is just created in its most simple declaration. It does not take any parameters, or arguements. So if you have a simple class, that does not have many user dependent variables and fields, that does a rather general task, then it is better to use default constructors, where you do not have to assign any user variables to class variables, and have to just call some basic intantiating functions depending on class requirements and functions.

   (b) **Parameterized Constructor**: Say there are variables that the user has entered that need to be assigned to the class object, or there are certain properties of each object different from other objects of the same class, like in enemies in a game, or employees in a Company, each object can be initialized with a set of variables. In this situation it is better to just use a parameterized constructor.

   (c) **Copy Constructor**: If you have many constructors that are often similar in defintion and declaration, but have very few dissimilar properties, it is better to use copy constructors. For example Trees in a RPG game, where each tree has the same basic structure, but you might have small variation in just the height or the position of the tree.

4. **Explain use of static member functions.**

Static member functions in java are those that can be accessed by other classes without declaring an Object of that class. This is often why the main function needs to be public and static. Every other member function needs to be accessed by an obejct of that class, as opposed to static ones.

In terms of memory, the *static* keyword in C++ is used when you have a variable that needs to be accessed by several objects of the same class, and this variable doesnt need to be different for each object. An Example would be the Security number of an Employee, which just needs to be incremented as an object is created. It is accessed by each object, and therefore it makes sense for it to be declared in a way where it does not get copied for each object, thereby saving space.

5. **How java program is executed?**
   Java, being a *platform-independent programming language*, doesn't work on the one-step compilation. Instead, it involves a two-step execution, first through an OS-independent compiler; and second, in a virtual machine (JVM) which is custom-built for every operating system. First, the source '.java' file is passed through the compiler, which then encodes the source code into a machine-independent encoding, known as Bytecode. The content of each class contained in the source file is stored in a separate '.class' file.

   The class files generated by the compiler are independent of the machine or the OS, which allows them to be run on any system. To run, the main class file (the class that contains the method main) is passed to the JVM and then goes through three main stages before the final machine code is executed.

6. **What is the use of JVM?**
   Java Virtual Machine, or JVM, loads, verifies and executes Java bytecode. It is known as the interpreter or the core of Java programming language because it executes Java programming.

   JVM is specifically responsible for converting bytecode to machine-specific code and is necessary in both JDK and JRE. It is also platform-dependent and performs many functions, including memory management and security. In addition, JVM can run programs written in other programming languages that have been translated to Java bytecode.

7. **What are the different control statements used in C++ and Java?**
   There are several Control statements in Java and C++

   (a) Loops like for, while and do while

   (b) Logical Control statements like if, else if, else, switch statements

   (c) Ternary Operator as another form of logical operation

   (d) Branching Statements include Keywords like break and continue.

   Examples:

```
1     // Loops
2     for(int i = 0; i < 5; i++)
3     {
4       cout<<"This is a basic for loop, and exists in both cpp and java.";
5     }
6     do
7     {
8       cout<<"This is a do while loop";
```

```
9     }while(condition);
10    while(true)
11    {
12      cout<<"This is a while loop";
13    }
14
15    // Logical Statements
16    if(condition)
17    {
18      cout<<"Condition true";
19    }
20    else cout << "Condition false";
21
22    condition = condition ? true : false;
23
24    // Branching statements:
25    switch(condition)
26    {
27      case 1: cout<<"Case 1 is being executed";
28        break;
29      case 2: cout<<"Case 2 is being executed";
30        break;
31      default: cout<<"Default case";
32        break;
33    }
34
```

8. **Write couple of examples/applications suitable to use OOP concepts specially use of classes, objects and constructors.**

   (a) **Game Development**: Enemies, walls, obstacles, trees, NPCs, are often structured as classes. This is because they have a set template that each member follows, and there are often many of them in a game. This makes OOP the perfect choice.

   (b) **Machine Learning**: Machine learning often requires extensive and complex algorithms that need to be written and applied on a set of data. If such algorithms are put together as classes, then their objects can be fed that data and that algorithm can be run on it efficiently and easily, as opposed to writing it every time for each data set.

   (c) **Software Development**: GUI components like buttons, sliders, panels, frames, bars, etc often have a singular functionality associated with them. As there are many such components in a GUI, it makes sense to make them into classes, and spawn their objects in various meaningful positions in the UI.

MIT World Peace University

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 1

Implementation of Inheritance using C++ and Java

Practical Report

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

November 29, 2022

Contents

1 Aim and Objectives

Aim

To Implement inheritance using C++ and Java (with interfaces).

Objectives

1. To understand the inheritance or is-A relationship concept.
2. To understand code reusability.
3. To learn implementation of interfaces in java.

2 Problem Statements

2.1 Problem 1 in C++

Design and develop inheritance for a given case study, identify objects and relationships and implement inheritance wherever applicable using C++.
Employee class has Empname, Empid, Address, Mailid, and Mobileno as data members. Add the Following Classes

- Programmer
- Team Leader
- Assistant Project Manager
- Project Manager from Employee Class

Add Basic Pay as the member of all the inherited classes with 97 % of the Basic Pay as DA, 10 % of Basic Pay as HRA, 12 % of Basic Pay as PF, 0.1 % of Basic Pay for staff club fund.

Generate Pay slips for the Employees with their gross and net salaries.

2.2 Problem 2 in Java

Write a Java Program for demonstrating Inheritance in Java. Write a program in Java showing hierarchical inheritance with base class as Employee and derived classes as FullTimeEmployee and InternEmployee with methods DisplaySalary in base class and CalculateSalary in derived classes. Calculate salary method will calculate as per increment given to fulltime and intern Employees. Fulltime employee- 50% hike, Intern employee-25% hike. Display salary before and after hike.

2.3 Probelm 3 in Java

Write a java program to create two interfaces Motorbike and Cycle.

- Motorbike interface consists of the attribute speed.
- The method is totalDistance().
- Cycle interface consists of the attributes distance and the method speed().
- These interfaces are implemented by the class TwoWheeler.
- Calculate total distance travelled and Average Speed maintained by Two Wheeler.

3 Theory

3.1 Concept of Inheritance and its types

Inheritance is a process in which one object acquires all the properties and behaviors of its parent object automatically. In such way, you can *reuse, extend, or modify* the attributes and behaviors which are defined in other class.

In C++, the class which inherits the members of another class is called derived class and the class whose members are inherited is called base class. The derived class is the specialized class for the base class. Most ideas of inheritance are directly applicable in Java as well.

3.1.1 Advantages

1. Minimizing duplicate code: Key benefits of Inheritance include minimizing the identical code as it allows sharing of the common code among other subclasses.
2. Flexibility: Inheritance makes the code flexible to change, as you will adjust only in one place, and the rest of the code will work smoothly.
3. Overriding: With the help of Inheritance, you can override the methods of the base class.
4. Data Hiding: The base class in Inheritance decides which data to be kept private, such that the derived class will not be able to alter it.

3.1.2 Types of Inheritance

1. Single inheritance : It is defined as the inheritance in which a derived class is inherited from the only one base class.
2. Multiple inheritance : It is the process of deriving a new class that inherits the attributes from two or more classes.
3. Hierarchical inheritance : When one class inherits another class which is further inherited by another class, it is known as multi level inheritance in C++. Inheritance is transitive so the last derived class acquires all the members of all its base classes.
4. Multilevel inheritance: It is a process of deriving a class from another derived class.
5. Hybrid inheritance : Any legal combination of any of the above inheritance techniques would be known as hybrid inheritance.

Let us Look an Examples of these.

```
class A
{
  // Base Class
};

class D
{
  // Base Class 2
}
class B : public A
{
  // Single Inheritance
```

```
  }
  class C : public B
  {
    // Multi Level Inheritance
  }
  class E : public A, public D
  {
    // Multiple Inheritance
  }
  class F : public E, private D
  {
    // hierarchical inheritance
  }
```

3.1.3 Base class and Derived Class Constructors

Whenever we create an object of a class, the default constructor of that class is invoked automatically to initialize the members of the class.

If we inherit a class from another class and create an object of the derived class, it is clear that the default constructor of the derived class will be invoked but before that the default constructor of all of the base classes will be invoke, i.e the order of invocation is that the base class's default constructor will be invoked first and then the derived class's default constructor will be invoked.

When a class is inherited from other, The data members and member functions of base class comes automatically in derived class based on the access specifier but the definition of these members exists in base class only. So when we create an object of derived class, all of the members of derived class must be initialized but the inherited members in derived class can only be initialized by the base class's constructor as the definition of these members exists in base class only.

This is why the constructor of base class is called first to initialize all the inherited members.

Let us see an Example

```

// base class
class Parent
{
    public:

    // base class constructor
    Parent()
    {
        cout << "Inside base class" << endl;
    }
};

// sub class
class Child : public Parent
{
    public:

    //sub class constructor
    Child()
```

```
    {
        cout << "Inside sub class" << endl;
    }
};

// main function
int main() {

    // creating object of sub class
    Child obj;

    return 0;
}

```

Output would be

```
Inside base class
Inside sub class
```

4 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers : g++ and gcc on linux for C++, and javac, with JDK 18.0.2 for Java

5 Input

For C++

1. Number of Each Type of Employee
2. Name, Age, Address City, and Salary of Each Employee

For Java

1. The Information and Salary about the Full Time Employee
2. The Information and Salary of the Intern Employee
3. Speed, Time and Distance

6 Output

For C++

1. General Information about Each Employee
2. The Gross Salary of Each Employee
3. The Net Salary of Each Employee

For Java

1. The General information about the Full time and the Intern Employee
2. The Hiked salaries of both the Intern Employee, and the Full Time Employee.
3. Speed and Distance.

7 Code

7.1 C++ Implementation for Problem 1

```
// Design and develop inheritance for a given case study, identify objects and relationships
// and implement inheritance wherever applicable using C++.
// Employee class has Emp_name, Emp_id, Address, Mail_id, and Mobile_no as data
// members.

// Inherit the classes:
// Programmer
// Team Lead
// Assistant Project Manager and
// Project Manager from employee class.

// Add Basic Pay as the member of all the inherited classes with 97% of Basic Pay as DA, 10
// % of Basic Pay as HRA, 12% of Basic Pay as PF, 0.1% of Basic Pay for staff club fund.
// Generate pay slips for the employees with their gross and net salary.

#include <iostream>
using namespace std;

class Employee
{

protected:
    static int ssn;
    int emp_id = 1000;
    int age = 0;
    double basic_sal = 0, da = 0, ta = 0, gross_sal = 0, net_sal = 0;
    string address_city, position, name;

public:
    // Default Constructor
    Employee()
    {
        cout << "The Default Constructor was called" << endl;
    }

    // Parameterized Constructor
    Employee(int e, int a, string add, string nam)
    {
        cout << "Parameterized constructor was called\n";
        emp_id = e;
        age = a;
        address_city = add;
```

```
43          name = nam;
44      }
45
46      // Copy Constructor
47      Employee(Employee &E)
48      {
49          cout << "Copy Constructor was called" << endl;
50          emp_id = E.emp_id;
51          age = E.age;
52          address_city = E.address_city;
53          name = E.name;
54      }
55
56      void display()
57      {
58          Employee::ssn++;
59          cout << "Employee ssn is:" << ssn << endl;
60          cout << "Employee ID is : " << emp_id << endl;
61          cout << "Employee Name: " << name << endl;
62          cout << "Employee Age: " << age << endl;
63          cout << "Employee Address City: " << address_city << endl;
64      }
65
66      void accept()
67      {
68          cout << "Enter the Employee ID: " << endl;
69          cin >> emp_id;
70          cout << "Enter the Employee Name: " << endl;
71          cin >> name;
72          cout << "Enter the Employee Age: " << endl;
73          cin >> age;
74          cout << "Enter the Employee Address City: " << endl;
75          cin >> address_city;
76      }
77
78      // Destructor
79      ~Employee()
80      {
81          cout << "The Destructor was called" << endl;
82      }
83  };
84
85  int Employee::ssn = 1000;
86
87  class Programmer : public Employee
88  {
89
90  protected:
91      double da = 0, hra = 0, pf = 0, scf = 0;
92
93  public:
94      void calc_gross_sal()
95      {
96          da = 0.97 * basic_sal;
97          hra = basic_sal;
98          pf = basic_sal;
99          scf = basic_sal;
100         gross_sal = da + hra + pf + scf + basic_sal;
101     }
```

```
102
103     void calc_net_sal()
104     {
105         // Reducing Income Taxes
106         net_sal = gross_sal - (0.15) * gross_sal;
107     }
108
109     void accept()
110     {
111         Employee::accept();
112         cout << "Enter the basic Salary of the Programmer : " << endl;
113         cin >> basic_sal;
114         calc_gross_sal();
115         calc_net_sal();
116     }
117
118     void display()
119     {
120         Employee::display();
121         cout << "The Gross Salary is: " << gross_sal << endl;
122         cout << "The Net Salary is: " << net_sal << endl;
123     }
124 };
125
126 class TeamLeader : public Employee
127 {
128
129 protected:
130     double da = 0, hra = 0, pf = 0, scf = 0;
131
132 public:
133     void calc_gross_sal()
134     {
135         da = 0.97 * basic_sal;
136         hra = basic_sal;
137         pf = basic_sal;
138         scf = basic_sal;
139         gross_sal = da + hra + pf + scf + basic_sal;
140     }
141
142     void calc_net_sal()
143     {
144         // Reducing Income Taxes
145         net_sal = gross_sal - (0.15) * gross_sal;
146     }
147
148     void accept()
149     {
150         Employee::accept();
151         cout << "Enter the basic Salary of the Team Leader : " << endl;
152         cin >> basic_sal;
153         calc_gross_sal();
154         calc_net_sal();
155     }
156
157     void display()
158     {
159         Employee::display();
160         cout << "The Gross Salary is: " << gross_sal << endl;
```

```
        cout << "The Net Salary is: " << net_sal << endl;
    }
};

class AssistantProjectManager : public Employee
{

protected:
    double da = 0, hra = 0, pf = 0, scf = 0;

public:
    void calc_gross_sal()
    {
        da = 0.97 * basic_sal;
        hra = basic_sal;
        pf = basic_sal;
        scf = basic_sal;
        gross_sal = da + hra + pf + scf + basic_sal;
    }

    void calc_net_sal()
    {
        // Reducing Income Taxes
        net_sal = gross_sal - (0.15) * gross_sal;
    }

    void accept()
    {
        Employee::accept();
        cout << "Enter the basic Salary of the Assistant Project Manager :" <<
    endl;
        cin >> basic_sal;
        calc_gross_sal();
        calc_net_sal();
    }

    void display()
    {
        Employee::display();
        cout << "The Gross Salary is: " << gross_sal << endl;
        cout << "The Net Salary is: " << net_sal << endl;
    }
};

class ProjectManager : public Employee
{

protected:
    double da = 0, hra = 0, pf = 0, scf = 0;

public:
    void calc_gross_sal()
    {
        da = 0.97 * basic_sal;
        hra = basic_sal;
        pf = basic_sal;
        scf = basic_sal;
        gross_sal = da + hra + pf + scf + basic_sal;
    }
```

```
219
220     void calc_net_sal()
221     {
222         // Reducing Income Taxes
223         net_sal = gross_sal - (0.15) * gross_sal;
224     }
225
226     void accept()
227     {
228         Employee::accept();
229         cout << "Enter the basic Salary of the Project Manager :" << endl;
230         cin >> basic_sal;
231         calc_gross_sal();
232         calc_net_sal();
233     }
234
235     void display()
236     {
237         Employee::display();
238         cout << "The Gross Salary is: " << gross_sal << endl;
239         cout << "The Net Salary is: " << net_sal << endl;
240     }
241 };
242
243 int main()
244 {
245     cout << "Welcome to Employee Payroll Management System" << endl
246          << endl;
247
248     int choice = 1, number = 1;
249
250     do
251     {
252         cout << "\n\nWhose Details do you wanna enter? " << endl;
253         cout << "1. Programmer\n2. Team Leader\n3. Assistant Project Manager\n4.
    Project Manager\n5. Quit\n";
254         cin >> choice;
255
256         if (choice == 1)
257         {
258             cout << "How many Programmers are we talking? ";
259             cin >> number;
260             Programmer pr[number];
261             for (int i = 0; i < number; i++)
262             {
263                 cout << "Enter the Information about the Programmer" << endl;
264                 pr[i].accept();
265             }
266             cout << "\nHere is their Information and their Pay Slips" << endl;
267             cout << endl
268                  << endl;
269
270             cout << "Programmer" << endl;
271
272             for (int i = 0; i < number; i++)
273             {
274                 cout << "Info and Pay Slip of Programmer " << i + 1 << endl;
275                 pr[i].display();
276                 cout << endl;
```

```
277             }
278         }
279         else if (choice == 2)
280         {
281             cout << "How many Team Leaders are we talking? ";
282             cin >> number;
283             TeamLeader tl[number];
284             for (int i = 0; i < number; i++)
285             {
286                 cout << "Enter the Information about the Team Leader " << i + 1 <<
    endl;
287                 tl[i].accept();
288             }
289             cout << "Here is their Information and their Pay Slips" << endl;
290             cout << endl
291                  << endl;
292             for (int i = 0; i < number; i++)
293             {
294                 cout << "Info and Pay Slip of Team Leader " << i + 1 << endl;
295                 tl[i].display();
296                 cout << endl;
297             }
298             cout << endl
299                  << endl;
300         }
301         else if (choice == 3)
302         {
303             cout << "How many Assistant Project Managers are we talking? ";
304             cin >> number;
305             AssistantProjectManager ap[number];
306             for (int i = 0; i < number; i++)
307             {
308                 cout << "Enter the Information about the Assitant Project Manager
    " << i + 1 << endl;
309                 ap[i].accept();
310             }
311             cout << "Here is their Information and their Pay Slips" << endl;
312             cout << endl
313                  << endl;
314             for (int i = 0; i < number; i++)
315             {
316                 cout << "Info and Pay Slip of Assitant Project Manager " << i + 1
    << endl;
317                 ap[i].display();
318                 cout << endl;
319             }
320             cout << endl
321                  << endl;
322         }
323 
324         else if (choice == 4)
325         {
326             cout << "How many Project Managers are we talking? ";
327             cin >> number;
328             ProjectManager pm[number];
329             for (int i = 0; i < number; i++)
330             {
331                 cout << "Enter the Information about the Project Manager " << i +
    1 << endl;
```

```
                pm[i].accept();
            }
            cout << "Here is their Information and their Pay Slips" << endl;
            cout << endl
                 << endl;
            for (int i = 0; i < number; i++)
            {
                cout << "Info and Pay Slip of Project Manager " << i + 1 << endl;
                pm[i].display();
                cout << endl;
            }
        }
    } while (choice != 5);

    return 0;
}
```

Listing 1: Main.Cpp

7.1.1 C++ Input and Output

```

Welcome to Employee Payroll Management System

Whose Details do you wanna enter?
1. Programmer
2. Team Leader
3. Assistant Project Manager
4. Project Manager
5. Quit
1
How many Programmers are we talking? 1
The Default Constructor was called
Enter the Information about the Programmer
Enter the Employee ID:
1001
Enter the Employee Name:
Tony
Enter the Employee Age:
45
Enter the Employee Address City:
Berlin
Enter the basic Salary of the Programmer :
450000

Here is their Information and their Pay Slips


Programmer
Info and Pay Slip of Programmer 1
Employee ssn is:1001
Employee ID is : 1001
Employee Name: Tony
Employee Age: 45
Employee Address City: Berlin
The Gross Salary is: 2.2365e+06
The Net Salary is: 1.90102e+06

The Destructor was called
```

```
39
40
41 Whose Details do you wanna enter?
42 1. Programmer
43 2. Team Leader
44 3. Assistant Project Manager
45 4. Project Manager
46 5. Quit
47 2
48 How many Team Leaders are we talking? 1
49 The Default Constructor was called
50 Enter the Information about the Team Leader 1
51 Enter the Employee ID:
52 1002
53 Enter the Employee Name:
54 Steve
55 Enter the Employee Age:
56 80
57 Enter the Employee Address City:
58 Queens
59 Enter the basic Salary of the Team Leader :
60 70000
61 Here is their Information and their Pay Slips
62
63
64 Info and Pay Slip of Team Leader 1
65 Employee ssn is:1002
66 Employee ID is : 1002
67 Employee Name: Steve
68 Employee Age: 80
69 Employee Address City: Queens
70 The Gross Salary is: 347900
71 The Net Salary is: 295715
72
73
74
75 The Destructor was called
76
77
78 Whose Details do you wanna enter?
79 1. Programmer
80 2. Team Leader
81 3. Assistant Project Manager
82 4. Project Manager
83 5. Quit
84 3
85 How many Assistant Project Managers are we talking? 1
86 The Default Constructor was called
87 Enter the Information about the Assitant Project Manager 1
88 Enter the Employee ID:
89 1003
90 Enter the Employee Name:
91 Caulson
92 Enter the Employee Age:
93 60
94 Enter the Employee Address City:
95 Delhi
96 Enter the basic Salary of the Assistant Project Manager :
97 60000
```

```
Here is their Information and their Pay Slips


Info and Pay Slip of Assitant Project Manager 1
Employee ssn is:1003
Employee ID is : 1003
Employee Name: Caulson
Employee Age: 60
Employee Address City: Delhi
The Gross Salary is: 298200
The Net Salary is: 253470



The Destructor was called


Whose Details do you wanna enter?
1. Programmer
2. Team Leader
3. Assistant Project Manager
4. Project Manager
5. Quit
4
How many Project Managers are we talking? 1
The Default Constructor was called
Enter the Information about the Project Manager 1
Enter the Employee ID:
1005
Enter the Employee Name:
Fury
Enter the Employee Age:
56
Enter the Employee Address City:
Pune
Enter the basic Salary of the Project Manager :
600000
Here is their Information and their Pay Slips


Info and Pay Slip of Project Manager 1
Employee ssn is:1004
Employee ID is : 1005
Employee Name: Fury
Employee Age: 56
Employee Address City: Pune
The Gross Salary is: 2.982e+06
The Net Salary is: 2.5347e+06

The Destructor was called


Whose Details do you wanna enter?
1. Programmer
2. Team Leader
3. Assistant Project Manager
4. Project Manager
5. Quit
```

```
5
```

Listing 2: C++ Output

7.2 Java Implementation of Problem 2

```java
package assignment_2;
import java.util.Scanner;

class Employee
{
    Scanner input = new Scanner(System.in);
    String name;
    int emp_id;
    double salary;
    int hike = 0;

    final static int full_time_hike_perc = 100;
    final static int intern_hike_perc = 50;

    void accept()
    {
        System.out.println("Enter the Employee Name");
        name = input.next();
        System.out.println("Enter The Employee ID");
        emp_id = input.nextInt();
        System.out.println("Enter The Employee Salary");
        salary = input.nextInt();
    }

    void display_salary()
    {
        System.out.println("The Employee Name is: " + name);
        System.out.println("The Employee ID is: " + emp_id);
    }
}
```

Listing 3: Employee.java

```java
package assignment_2;

class InternEmployee extends Employee {
    double hiked_salary = 0;

    void calculate_salary() {
        hiked_salary = salary + salary * (full_time_hike_perc / 100);
    }

    @Override
    void display_salary() {
        super.display_salary();
        System.out.println("The Salary Before the Hike for this Intern Employee is
    : " + salary);
        calculate_salary();
        System.out.println("The Salary after the Hike for this Intern Employee is
    : " + hiked_salary);
    }
```

```
}
```

Listing 4: Intern Employee.java

```
package assignment_2;

class FullTimeEmployee extends Employee {
    double hiked_salary = 0;

    void calculate_salary() {
        hiked_salary = salary + salary * (full_time_hike_perc / 100);
    }

    @Override
    void display_salary() {
        super.display_salary();
        System.out.println("The Salary Before the Hike for this Full time Employee
     is: " + salary);
        calculate_salary();
        System.out.println("The Salary after the Hike for this Full time Employee
    is : " + hiked_salary);
    }
}
```

Listing 5: Full Time Employee.java

```

// Write a Java Program for demonstrating Inheritance in Java.
// Write a program in Java showing hierarchical inheritance with base class as
    Employee and
// derived classes as FullTimeEmployee and InternEmployee with methods
    DisplaySalary in
// base class and CalculateSalary in derived classes.
// Calculate salary method will calculate as per increment given to fulltime and
    intern
// Employees. Fulltime employee- 50% hike, Intern employee-25% hike. Display
    salary
// before and after hike.


package assignment_2;

import java.util.Scanner;

class Main {
    public static void main(String args[]) {
        System.out.println("Welcome to Salary Hiking Program");
        Employee emp = new Employee();
        FullTimeEmployee full_time_emp = new FullTimeEmployee();
        System.out.println("Enter the Details about the Full Time Employee: \n");
        full_time_emp.accept();
        full_time_emp.display_salary();
        InternEmployee intern_emp = new InternEmployee();
        System.out.println("Enter the Details about the Intern Employee: \n");
        intern_emp.accept();
        intern_emp.display_salary();
    }
}
```

Listing 6: Main.java

7.2.1 Java Output for Problem 2

```
Welcome to Salary Hiking Program
Enter the Details about the Full Time Employee:

Enter the Employee Name
Tony
Enter The Employee ID
1001
Enter The Employee Salary
100000
The Employee Name is: Tony
The Employee ID is: 1001
The Salary Before the Hike for this Full time Employee is: 100000.0
The Salary after the Hike for this Full time Employee is : 200000.0
Enter the Details about the Intern Employee:

Enter the Employee Name
Steve
Enter The Employee ID
1002
Enter The Employee Salary
50000
The Employee Name is: Steve
The Employee ID is: 1002
The Salary Before the Hike for this Intern Employee is: 50000.0
The Salary after the Hike for this Intern Employee is : 100000.0
```

Listing 7: Java Output for Problem 2

7.3 Java Implementation of Problem 3 using Interfaces

```
package assignment_2b;

import java.util.Scanner;

public class TwoWheeler implements MotorCycle, Cycle {
    public int speed = 0;
    public int time = 0;
    public int total_distance = 0;

    @Override
    public void speed() {
        System.out.println(total_distance / time);
    }

    @Override
    public void totalDistance() {
        System.out.println(speed * time);
    }
}
```

Listing 8: Two Wheeler.java

```
package assignment_2b;

public interface MotorCycle {
    public abstract void totalDistance();
```

```
}
```

Listing 9: MotorCycle.java

```
package assignment_2b;

public interface Cycle {
    public abstract void speed();
}
```

Listing 10: Cycle.java

```
// Write a java program to create two interfaces Motorbike and Cycle.
// Motorbike interface consists of the attribute speed.
// The method is totalDistance().
// Cycle interface consists of the attributes distance and the method speed().
// These interfaces are implemented by the class TwoWheeler.
// Calculate total distance travelled and Average Speed maintained by Two Wheeler.

package assignment_2b;

import java.util.Scanner;

public class Main {

    public static void main(String args[]) {
        Scanner input = new Scanner(System.in);
        TwoWheeler obj = new TwoWheeler();

        System.out.println("Calculating Speed: ");
        System.out.println("Enter the Distance Travelled by Your 2 Wheeler: ");
        obj.total_distance = input.nextInt();

        System.out.println("Enter the Time you Travelled on Your 2 Wheeler: ");
        obj.time = input.nextInt();

        System.out.println("The Speed is: ");
        obj.speed();

        System.out.println("Calculating Total Distance: ");
        System.out.println("Enter the Speed of Your 2 Wheeler: ");
        obj.speed = input.nextInt();

        System.out.println("Enter the Time you Travelled on Your 2 Wheeler: ");
        obj.time = input.nextInt();

        System.out.println("The Total Distance is: ");
        obj.totalDistance();
    }
}
```

Listing 11: Main.java

7.3.1 Java Output

```
Calculating Speed:
Enter the Distance Travelled by Your 2 Wheeler:
50
Enter the Time you Travelled on Your 2 Wheeler:
2
```

```
The Speed is:
25
Calculating Total Distance:
Enter the Speed of Your 2 Wheeler:
25
Enter the Time you Travelled on Your 2 Wheeler:
2
The Total Distance is:
50
```

Listing 12: Java Output for Program 3

8 Conclusion

Thus, learned to use reusability by applying concept of inheritance, interfaces and implemented solution of the given problem statement using C++ and Java.

9 FAQs

1. **Discuss ambiguity arises in multipath inheritance and how it is to be avoided in C++?**
   In *multiple* inheritances, when one class is derived from two or more base classes then there may be a possibility that the base classes have functions with the same name, and the derived class may not have functions with that name as those of its base classes.

   If the derived class object needs to access one of the similarly named member functions of the base classes then it results in ambiguity because the compiler gets confused about which base's class member function should be called.

   To solve this ambiguity scope resolution operator is used denoted by ' :: '

   The Syntax to use it is:

```
    ObjectName.ClassName::FunctionName();

```

   An Example:

```
// C++ program to resolve inheritance
// ambiguity

#include<iostream>
using namespace std;

// Base class A

class A {
    public:

    void func() {
        cout << " I am in class A" << endl;
    }
};

// Base class B

class B {
    public:

    void func() {
        cout << " I am in class B" << endl;
    }
};

// Derived class C
class C: public A, public B {


};
```

```

// Driver Code

int main() {

    // Created an object of class C
    C obj;

    // Calling function func() in class A
    obj.A::func();

    // Calling function func() in class B
    obj.B::func();

    return 0;
}


```

2. **What's the difference between public, private, and protected?**
   public, private, and protected are known as access modifiers. Like their name suggests, they modify the access given to objects and subclasses.

   (a) If you do class child : **private** parent; then every private data member becomes inaccessible, coz anyway thats what should happen, then the protected data members become private, and public data members also become private.

   (b) If you do class child : **protected** parent; then its the same thing, except you still cant access private variables, but protected and public data members become protected

   (c) Same with class child : **public** parent; everything remains unchanged. The objects will behave in accordance with the usual laws of objects.

3. **Why can't derived class access private things from base class?**:
   The **private** access modifier, when used in any class, by its definition restricts sub classes and its objects to access variables declared in its scope. As a result, derived classes can not access the private variables defined in the base class.

```
    class A
    {
      private:
        int a;
    };
    class B: public A
    {

    }obj;
    int main()
    {
      A.a // not accessible
    }

```

4. **Explain use of 'super' keyword with suitable example**
   The *super* keyword in Java refers to the parent class's variables and functions. Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

   - super can be used to refer immediate parent class instance variable.
   - super can be used to invoke immediate parent class method.
   - super() can be used to invoke immediate parent class constructor.

5. **Why to use concept of interface in Java**

   An **Interface** in Java programming language is defined as an abstract type used to specify the behavior of a class. An interface in Java is a blueprint of a class. A Java interface contains static constants and abstract methods.
   The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not the method body. It is used to achieve abstraction and multiple inheritance in Java. In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body. Java Interface also represents the IS-A relationship.

   Like a class, an interface can have methods and variables, but the methods declared in an interface are by default abstract (only method signature, no body).

   - **Provides Communication** - One of the uses of the interface is to provide communication. Through interface you can specify how you want the methods and fields of a particular type.
   - **Multiple Inheritance** - Java doesn't support multiple inheritance, using interfaces you can achieve multiple inheritance.
   - **Abstraction**- Abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user. In other words, the user will have the information on what the object does instead of how it does it.
     Since all the methods of the interface are abstract and user doesn't know how a method is written except the method signature/prototype. Using interfaces, you can achieve (complete) abstraction.
   - **Loose Coupling** : Coupling refers to the dependency of one object type on another, if two objects are completely independent of each other and the changes done in one doesn't affect the other both are said to be loosely coupled. You can achieve loose coupling in Java using interfaces

6. **Write Couple of Examples or Applications suitable to Demonstrate use of Inheritances.**

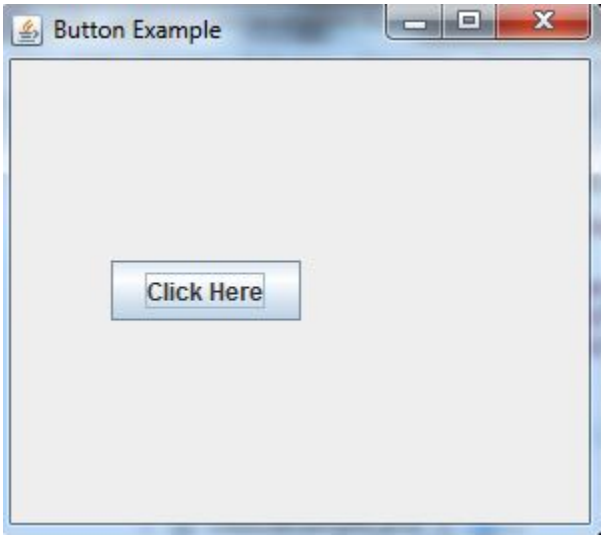
   Inheritance is a core part of Object Oriented Programming. In every Object oriented language, it is heavily used, and these languages are heavily used in almost all fields involving programming, which makes the uses of inheritance vast and versatile. So some of them are listed below.

(a) In Application Development: Every GUI that we see on screens, is often built on simple base classes like buttons, sliders, labels, titlebars etc. The Programmer often just inherits it, and overrides certain methods to his or her liking.

(b) Games are built in many different ways, but as the scale of games increase it becomes pertinent to use simple base classes like trees, enemies, players, friends, coins, As there might be several of the same kind of enemies, and several sorts of trees in the game environment, which will all inherit the base class and override or create certain new methods.

(c) In Data Science, Many algorithms are their own predefined classes, The programmer then tweaks certain parameters, and this helps them streamline and organize the process of applying this algorithm on hundreds of different datasets without hassle.

```
// Here is an example of Creating a GUI in Java using Swing.

// This is the definition of the JButton class, which is used for placing buttons on the screen.
public class JButton extends AbstractButton implements Accessible


import javax.swing.*;
public class ButtonExample {
  public static void main(String[] args) {

    JFrame f=new JFrame("Button Example");
    JButton b=new JButton("Click Here");

    b.setBounds(50,100,95,30);
    f.add(b);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
  }
}


```



MIT World Peace University

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 1

Implementation of Polymorphism using C++ and JAVA

Practical Report

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

November 29, 2022

Contents

1 Aim and Objectives

Aim

Implementation of Polymorphism using C++ and Java.

Objectives

1. To understand the use of pure virtual funcitons.
2. To understnad implantation of compile time and run time polymorphism.
3. To learn implementation ofmethod overriding in java.

2 Problem Statements

2.1 Problem 1 in C++

Write a C++ program with base class Employee and three derived classes namely

- SalariedEmployees
- CommissionEmployees
- HourlyEmployees

Declare calculateSalary() as a pure virtual function in base class and define it in respective derived classes to calculate salary of an employee. The company wants to implement an Object Oriented Application that performs its payroll calculations polymorphically.

2.2 Problem 2 in Java

Define a Class **Shapes** as the Base Class that can find the area of the following :

- Circle
- Square
- Rectangle

Find the area of these shapes using construtor overloading and method overloading.

2.3 Probelm 3 in Java

Create a Parent Class Hillstations with the methods location() and famousfor(). Create three subclasses by Hill Station names. These subclasses must extend the superclass and override its methods location() and famousfor(). It should refer to the base class object and the base class method overrides the superclass method, and the base class method is invoked at runtime.

3 Theory

3.1 Concept of Compile time Polymorphism

In compile-time polymorphism, a function is called at the time of program compilation. We call this type of polymorphism as early binding or Static binding.

Function overloading and operator overloading is the type of Compile time polymorphism.

It can be implemented in 2 simple ways in c++

1. Function Overloading Function overloading means one function can perform many tasks. In C++, a single function is used to perform many tasks with the same name and different types of arguments. In the function overloading function will call at the time of program compilation. It is an example of compile-time polymorphism.

```

    class Addition {
    public:
      int ADD(int X,int Y)   // Function with parameter
      {
        return X+Y;     // this function is performing addition of  two
    Integer value
      }
      int ADD() {               // Function with same name but without
    parameter
        string a= "HELLO";
        string b="SAM";   // in this function concatenation is performed
      string c= a+b;
      cout<<c<<endl;

      }
    };
    int main(void) {
      Addition obj;   // Object is created
      cout<<obj.ADD(128, 15)<<endl; //first method is called
      obj.ADD();  // second method is called
      return 0;
    }


```

2. Operator Overloading Operator overloading means defining additional tasks to operators without changing its actual meaning. We do this by using operator function.

   ‘’ The purpose of operator overloading is to provide a special meaning to the user-defined data types.

   The advantage of Operators overloading is to perform different operations on the same operand.

```
      #include <iostream>
      using namespace std;
      class A
      {

        string x;
        public:
        A(){}
        A(string i)
        {
        x=i;
```

```
12          }
13          void operator+(A);
14          void display();
15      };
16
17      void A:: operator+(A a)
18      {
19
20          string m = x+a.x;
21          cout<<"The result of the addition of two objects is : "<<m;
22
23      }
24      int main()
25      {
26          A a1("Welcome");
27          A a2("back");
28          a1+a2;
29          return 0;
30      }
31
32
```

3.2 Concept of Run Time Polymorphism

In Runtime polymorphism, functions are called at the time the program execution. Hence, it is known as late binding or dynamic binding.

Function overriding is a part of runtime polymorphism. In function overriding, more than one method has the same name with different types of the parameter list.

It is achieved by using virtual functions and pointers. It provides slow execution as it is known at the run time. Thus, It is more flexible as all the things executed at the run time.

In C++ it can be implemented using these 2 methods.

1. Function overriding: In function overriding, we give the new definition to base class function in the derived class. At that time, we can say the base function has been overridden. It can be only possible in the 'derived class'. In function overriding, we have two definitions of the same function, one in the superclass and one in the derived class. The decision about which function definition requires calling happens at runtime. That is the reason we call it 'Runtime polymorphism'.

2. Virtual Functions A virtual function is declared by keyword virtual. The return type of virtual function may be int, float, void.

   A virtual function is a member function in the base class. We can redefine it in a derived class. It is part of run time polymorphism. The declaration of the virtual function must be in the base class by using the keyword virtual. A virtual function is not static.

   The virtual function helps to tell the compiler to perform dynamic binding or late binding on the function.

   If it is necessary to use a single pointer to refer to all the different classes' objects. This is because we will have to create a pointer to the base class that refers to all the derived objects.

3.3 Use of Pure Virtual Functions

When the function has no definition, we call such functions as “Do-nothing function or Pure virtual function”. The declaration of this function happens in the base class with no definition.

4 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers : g++ and gcc on linux for C++, and javac, with JDK 18.0.2 for Java

5 Input

For C++

1. Number of Each Type of Employee
2. Name, Age, Address City, and Salary of Each Employee

For Java

1. The Side of the Square
2. The Radius of the Circle
3. The Length and Breadth of the Rectangle.

6 Output

For C++

1. General Information about Each Employee
2. The Weekly, hourly and commisioned Salary for Respective Employees.

For Java

1. The Area of the Shapes
2. The Location of the Hill Stations
3. The Reason the Hill stations are Famous for.

7 Code

7.1 C++ Implementation

```cpp
// Write a C++ program with base class Employee and three derived classes namely
// 1. salaried employees
// 2. commission_employees and
// 3. hourly employees
// Declare calculate_salary() as a pure virtual function in base class and define it in respective
// derived classes to calculate salary of an employee.
// The company wants to implement an Object Oriented Application that performs its payroll
// calculations polymorphically.

// Polymorphism
// Virtual Functionss, overriding functions, and overloading functions.

#include <iostream>
using namespace std;

class Employee
{
public:
    // static int ssn;
    int emp_id = 1000;
    int age = 0;
    double basic_sal = 0, da = 0, ta = 0, gross_sal = 0, net_sal = 0;
    string address_city, position, name;

    virtual void calculate_salary() = 0;

    void display()
    {
        // ssn++;
        // cout << "Employee ssn is:" << ssn << endl;
        cout << "Employee ID is : " << emp_id << endl;
        cout << "Employee Name: " << name << endl;
        cout << "Employee Age: " << age << endl;
        cout << "Employee Address City: " << address_city << endl;
    }

    void accept()
    {
        cout << "Enter the Employee ID: " << endl;
        cin >> emp_id;
        cout << "Enter the Employee Name: " << endl;
        cin >> name;
        cout << "Enter the Employee Age: " << endl;
        cin >> age;
        cout << "Enter the Employee Address City: " << endl;
        cin >> address_city;
    }

    // Destructor
    ~Employee()
    {
        cout << "The Destructor was called" << endl;
    }
};
class SalariedEmployee : public Employee
{
public:
```

```
58      int weekly_salary;
59      int net_sal;
60      void accept()
61      {
62          Employee::accept();
63          cout << "Enter the Wage: ";
64          cin >> weekly_salary;
65      }
66
67      void calculate_salary()
68      {
69          cout << "Calculating Salary of Salaried Employee" << endl;
70          net_sal = weekly_salary * 7;
71      }
72
73      void display()
74      {
75          Employee::display();
76          cout << "Weekly Salary is: " << net_sal << endl;
77      }
78  };
79
80  class HourlyEmployee : public Employee
81  {
82  public:
83      int hours, wage;
84      int net_sal;
85
86      void accept()
87      {
88          Employee::accept();
89          cout << "Enter the basic salary: " << endl;
90          cin >> basic_sal;
91          cout << "Enter the Wage: ";
92          cin >> wage;
93          cout << "Enter the hours worked" << endl;
94          cin >> hours;
95      }
96
97      void calculate_salary()
98      {
99          cout << "Calculating Salary of Salaried Employee" << endl;
100         if (hours < 40)
101         {
102             net_sal = basic_sal + hours * wage;
103         }
104         else
105         {
106             net_sal = 40 * wage + (hours - 40) * wage * 1.5;
107         }
108     }
109
110     void display()
111     {
112         Employee::display();
113         cout << "Hourly Employee Salary is: " << net_sal << endl;
114     }
115 };
116
```

```
117 class CommissionEmployee : public Employee
118 {
119 public:
120     float gross_sales, commission_rate = 0.05;
121     float net_sal;
122 
123     void accept()
124     {
125         Employee::accept();
126         cout << "Enter the gross sales: ";
127         cin >> gross_sales;
128     }
129 
130     void calculate_salary()
131     {
132         cout << "Calculating Salary of Salaried Employee" << endl;
133         net_sal = commission_rate * gross_sales;
134     }
135 
136     void display()
137     {
138         Employee::display();
139         cout << "Commission Employee Salary is: " << net_sal << endl;
140     }
141 };
142 
143 int main()
144 {
145     cout << "Welcome to Employee Payroll Management System" << endl
146          << endl;
147 
148     int choice = 1, number = 1;
149     Employee *ptr;
150     do
151     {
152         cout << "1. Salaried Employee\n2. Commisisioned Employee\n3. Hourly
    Employee\n4. Quit\n";
153         cout << "\n\nWhose Details do you wanna enter? " << endl;
154         cin >> choice;
155 
156         if (choice == 1)
157         {
158             cout << "How many SalariedEmployees are we talking? ";
159             cin >> number;
160             SalariedEmployee pr[number];
161             for (int i = 0; i < number; i++)
162             {
163                 cout << "Enter the Information about the Salaried Employee" <<
    endl;
164                 pr[i].accept();
165             }
166             cout << "\nHere is their Information and their Pay Slips" << endl;
167             cout << endl
168                  << endl;
169 
170             cout << "Salaried Employee" << endl;
171 
172             for (int i = 0; i < number; i++)
173             {
```

```
174                 cout << "Info and Pay Slip of Salaried Employee " << i + 1 << endl
    ;
175                 ptr = &pr[i];
176                 ptr->calculate_salary();
177                 pr[i].display();
178                 cout << endl;
179             }
180         }
181         else if (choice == 2)
182         {
183             cout << "How many Commission Employees are we talking? ";
184             cin >> number;
185             CommissionEmployee tl[number];
186             for (int i = 0; i < number; i++)
187             {
188                 cout << "Enter the Information about the Commission Employee " <<
    i + 1 << endl;
189                 tl[i].accept();
190             }
191             cout << "Here is their Information and their Pay Slips" << endl;
192             cout << endl
193                  << endl;
194             for (int i = 0; i < number; i++)
195             {
196                 cout << "Info and Pay Slip of Commission Employee " << i + 1 <<
    endl;
197                 ptr = &tl[i];
198                 ptr->calculate_salary();
199                 tl[i].display();
200                 cout << endl;
201             }
202             cout << endl
203                  << endl;
204         }
205         else if (choice == 3)
206         {
207             cout << "How many Hourly Employees are we talking? ";
208             cin >> number;
209             HourlyEmployee ap[number];
210             for (int i = 0; i < number; i++)
211             {
212                 cout << "Enter the Information about the Hourly Employees " << i +
     1 << endl;
213                 ap[i].accept();
214             }
215             cout << "Here is their Information and their Pay Slips" << endl;
216             cout << endl
217                  << endl;
218             for (int i = 0; i < number; i++)
219             {
220                 cout << "Info and Pay Slip of Hourly Employees" << i + 1 << endl;
221                 ptr = &ap[i];
222                 ptr->calculate_salary();
223                 ap[i].display();
224                 cout << endl;
225             }
226             cout << endl
227                  << endl;
228         }
```

```
229
230     } while (choice != 4);
231
232     return 0;
233 }
```

Listing 1: Main.Cpp

7.1.1 C++ Input and Output

```
1  Welcome to Employee Payroll Management System
2
3  1. Salaried Employee
4  2. Commisisioned Employee
5  3. Hourly Employee
6  4. Quit
7
8
9  Whose Details do you wanna enter?
10 1
11 How many SalariedEmployees are we talking? 1
12 Enter the Information about the Salaried Employee
13 Enter the Employee ID:
14 1001
15 Enter the Employee Name:
16 Tony
17 Enter the Employee Age:
18 35
19 Enter the Employee Address City:
20 NewYork
21 Enter the Wage: 50000
22
23 Here is their Information and their Pay Slips
24
25
26 Salaried Employee
27 Info and Pay Slip of Salaried Employee 1
28 Calculating Salary of Salaried Employee
29 Employee ID is : 1001
30 Employee Name: Tony
31 Employee Age: 35
32 Employee Address City: NewYork
33 Weekly Salary is: 350000
34
35 The Destructor was called
36 1. Salaried Employee
37 2. Commisisioned Employee
38 3. Hourly Employee
39 4. Quit
40
41 Whose Details do you wanna enter?
42 2
43 How many Commission Employees are we talking? 1
44 Enter the Information about the Commission Employee 1
45 Enter the Employee ID:
46 1002
47 Enter the Employee Name:
48 Steve
49 Enter the Employee Age:
```

```
105
Enter the Employee Address City:
Queens
Enter the gross sales: 500
Here is their Information and their Pay Slips


Info and Pay Slip of Commission Employee 1
Calculating Salary of Salaried Employee
Employee ID is : 1002
Employee Name: Steve
Employee Age: 105
Employee Address City: Queens
Commission Employee Salary is: 25



The Destructor was called
1. Salaried Employee
2. Commisisioned Employee
3. Hourly Employee
4. Quit


Whose Details do you wanna enter?
3
How many Hourly Employees are we talking? 1
Enter the Information about the Hourly Employees 1
Enter the Employee ID:
1003
Enter the Employee Name:
Bruce
Enter the Employee Age:
50
Enter the Employee Address City:
Space
Enter the basic salary:
600
Enter the Wage: 200
Enter the hours worked
45
Here is their Information and their Pay Slips


Info and Pay Slip of Hourly Employees1
Calculating Salary of Salaried Employee
Employee ID is : 1003
Employee Name: Bruce
Employee Age: 50
Employee Address City: Space
Hourly Employee Salary is: 9500



The Destructor was called
1. Salaried Employee
2. Commisisioned Employee
3. Hourly Employee
4. Quit
```

```
109
110
111 Whose Details do you wanna enter?
112 4
```

Listing 2: Output for Problem 1

7.2 Java Implementation of Problem 2

```
1  package assignment_3;
2
3  public class Shapes {
4
5      public double Area;
6      public double side;
7      public double length;
8      public double breadth;
9      public int radius;
10
11     Shapes(int radius) {
12         Area = 0.0;
13         this.radius = radius;
14     }
15
16     Shapes(double length, double breadth) {
17         Area = 0.0;
18         this.length = length;
19         this.breadth = breadth;
20     }
21
22     Shapes(double side) {
23         Area = 0.0;
24         this.side = side;
25     }
26
27     double Area(int radius) {
28         Area = 3.14 * radius * radius;
29         return Area;
30     }
31
32     double Area(double length, double breadth) {
33         Area = length * breadth;
34         return Area;
35     }
36
37     double Area(double side) {
38         Area = side * side;
39         return Area;
40     }
41
42 }
```

Listing 3: Full Time Employee.java

```
1 // Define a class Shapes as
2 // 1. Circle
3 // 2. square and
4 // 3. rectangle.
5 // Find the area of these shapes using constructor overloading and method
      overloading.
```

```java

package assignment_3;

import java.lang.Math;

public class Problem_A {
    public static void main(String[] args) {
        Shapes circle = new Shapes(7);
        Shapes square = new Shapes(1.5);
        Shapes rectangle = new Shapes(1.4, 3.5);
        System.out.println("The Radius of the Circle is: " + circle.radius);
        System.out.println("The Area of the Circle is: " + circle.Area(circle.
    radius));
        System.out.println("The Side of the Square is: " + square.side);
        System.out.println("The Area of the Square is: " + square.Area(square.side
    ));
        System.out.println("The Length of the Rectangle is: " + rectangle.length);
        System.out.println("The Breadth of the Rectangle is: " + rectangle.breadth
    );
        System.out.println("The Area of the Rectangle is: "
                + String.format("%.2f", rectangle.Area(rectangle.length, rectangle
    .breadth)));
    }
}
```

Listing 4: Main.java

7.2.1 Java Output for Problem 2

```
The Radius of the Circle is: 7
The Area of the Circle is: 153.86
The Side of the Square is: 1.5
The Area of the Square is: 2.25
The Length of the Rectangle is: 1.4
The Breadth of the Rectangle is: 3.5
The Area of the Rectangle is: 4.90
```

Listing 5: Output for Problem 2

7.3 Java Implementation of Problem 3 using Interfaces

```java
package assignment_3;

abstract public class HillStation {
    abstract public void location(); // Pure Virtual Function.

    abstract public void famousfor();
}

class Manali extends HillStation {
    @Override
    public void location() {
        System.out.println("Manali in Himachal Pradesh");
    }

    @Override
    public void famousfor() {
        System.out.println(
                "Manali is a high-altitude Himalayan resort town in India's
    northern Himachal Pradesh state. It has a reputation as a backpacking center.
```

```
    Set on the Beas River, it's a gateway for skiing in the Solang Valley and
    trekking in Parvati Valley. It's also a jumping-off point for paragliding,
    rafting and mountaineering in the Pir Panjal mountains, home to 4,000m-high
    Rohtang Pass.");
    }
}

class Shimla extends HillStation {
    @Override
    public void location() {
        System.out.println("Shimla is in Himachal Pradesh");
    }

    @Override
    public void famousfor() {
        System.out.println(
                "The town is famous for pleasant walking experiences on hillsides
    surrounded by pine and oak forests. This capital city of Himachal Pradesh is
    famous for The Mall, ridge, and toy train. With colonial style buildings, the
    town has relics of ancient past that lend it a distinct look.");
    }
}

class Mahabaleshwar extends HillStation {
    @Override
    public void location() {
        System.out.println("Mahabaleshwar is in Maharashtra");
    }

    @Override
    public void famousfor() {
        System.out.println(
                "Mahabaleshwar is a hill station in India's forested Western Ghats
     range, south of Mumbai. It features several elevated viewing points, such as
    Arthur's Seat. West of here is centuries-old Pratapgad Fort, perched atop a
    mountain spur. East, Lingmala Waterfall tumbles off a sheer cliff. Colorful
    boats dot Venna Lake, while 5 rivers meet at Panch Ganga Temple to the north.")
    ;
    }
}
```

Listing 6: HillStation

```
// Create a parent class Hillstations with the methods location() and famousfor().
// Create three subclasses by hill station names e.g. Manali, Shimla etc.
// Subclasses extend the superclass and override methods location() and famousfor
    ().
// Call the methods location() and famousfor() by the Parent class, i.e.
    Hillstations
// class.
// It should refer to the base class object and the base class method overrides
    the
// superclass method, base class method is invoked at runtime.

package assignment_3;

public class Problem_B {
    public static void main(String[] args) {
        Manali obj = new Manali();
```

```
        obj.location();
        obj.famousfor();

        Shimla obj1 = new Shimla();
        obj1.location();
        obj1.famousfor();

        Mahabaleshwar obj2 = new Mahabaleshwar();
        obj2.location();
        obj2.famousfor();
    }
}
```

Listing 7: Main.java

7.3.1 Java Output

```
Manali in Himachal Pradesh
Manali is a high-altitude Himalayan resort town in India's northern Himachal
    Pradesh state. It has a reputation as a backpacking center. Set on the Beas
    River, it's a gateway for skiing in the Solang Valley and trekking in Parvati
    Valley. It's also a jumping-off point for paragliding, rafting and
    mountaineering in the Pir Panjal mountains, home to 4,000m-high Rohtang Pass.
Shimla is in Himachal Pradesh
The town is famous for pleasant walking experiences on hillsides surrounded by
    pine and oak forests. This capital city of Himachal Pradesh is famous for The
    Mall, ridge, and toy train. With colonial style buildings, the town has relics
    of ancient past that lend it a distinct look.
Mahabaleshwar is in Maharashtra
Mahabaleshwar is a hill station in India's forested Western Ghats range, south of
    Mumbai. It features several elevated viewing points, such as Arthur's Seat.
    West of here is centuries-old Pratapgad Fort, perched atop a mountain spur.
    East, Lingmala Waterfall tumbles off a sheer cliff. Colorful boats dot Venna
    Lake, while 5 rivers meet at Panch Ganga Temple to the north.
```

Listing 8: Output for Problem 3

8 Conclusion

Thus, learned to use polymorphism and implemented solution of the given problem statement using C++ and Java.

9 FAQs

1. **Discuss the use of Virtual Functions.**
   Without "virtual" you get "early binding". Which implementation of the method is used gets decided at compile time based on the type of the pointer that you call through.

   With "virtual" you get "late binding". Which implementation of the method is used gets decided at run time based on the type of the pointed-to object - what it was originally constructed as. This is not necessarily what you'd think based on the type of the pointer that points to that object.

```
    class Base
    {
      public:
          void Method1 ()  {  std::cout << "Base::Method1" << std::endl;  }
      virtual void Method2 ()  {  std::cout << "Base::Method2" << std::endl;
    }
    };

    class Derived : public Base
    {
      public:
      void Method1 ()  {  std::cout << "Derived::Method1" << std::endl;  }
      void Method2 ()  {  std::cout << "Derived::Method2" << std::endl;  }
    };

    Base* basePtr = new Derived ();
      //  Note - constructed as Derived, but pointer stored as Base*

    basePtr->Method1 ();  //  Prints "Base::Method1"
    basePtr->Method2 ();  //  Prints "Derived::Method2"

```

2. **What is the difference between early binding and late binding.**
   Early Binding: The binding which can be resolved at compile time by the compiler is known as static or early binding. Binding of all the static, private and final methods is done at compile-time.

   Example

```
    public class NewClass {
      public static class superclass {
        static void print()
        {
          System.out.println("print in superclass.");
        }
      }
      public static class subclass extends superclass {
        static void print()
        {
          System.out.println("print in subclass.");
```

```
        }
    }

    public static void main(String[] args)
    {
        superclass A = new superclass();
        superclass B = new subclass();
        A.print();
        B.print();
    }
}

// Output
// print in superclass.
// print in superclass.


```

Late Binding: In the late binding or dynamic binding, the compiler doesn't decide the method to be called. Overriding is a perfect example of dynamic binding. In overriding both parent and child classes have the same method.

```


public class NewClass {
    public static class superclass {
        void print()
        {
            System.out.println("print in superclass.");
        }
    }

    public static class subclass extends superclass {
        @Override
        void print()
        {
            System.out.println("print in subclass.");
        }
    }

    public static void main(String[] args)
    {
        superclass A = new superclass();
        superclass B = new subclass();
        A.print();
        B.print();
    }
}
// Output:
// print in superclass.
// print in subclass.

```

3. **Explain the use of abstract keyword in java with examples.**
   Abstract is a non-access modifier in java applicable for classes, methods but not variables. It is used to achieve abstraction which is one of the pillar of Object Oriented Programming(OOP). Following are different contexts where abstract can be used in Java.

Due to their partial implementation, we cannot instantiate abstract classes.Any subclass of an abstract class must either implement all of the abstract methods in the super-class, or be declared abstract itself.Some of the predefined classes in java are abstract. They depend on their sub-classes to provide complete implementation. For example, java.lang.Number is a abstract class. For more on abstract classes, see abstract classes in java

```
// A java program to demonstrate
// use of abstract keyword.

// abstract with class
abstract class A
{
  // abstract with method
  // it has no body
  abstract void m1();

  // concrete methods are still allowed in abstract classes
  void m2()
  {
    System.out.println("This is a concrete method.");
  }
}

// concrete class B
class B extends A
{
  // class B must override m1() method
  // otherwise, compile-time exception will be thrown
  void m1() {
    System.out.println("B's implementation of m1.");
  }


}

// Driver class
public class AbstractDemo
{
  public static void main(String args[])
  {
    B b = new B();
    b.m1();
    b.m2();
  }
}


```

4. **State Features of abstract base classes.**
   A class which is declared as abstract is known as an abstract class. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

   Important Features are:

   (a) An abstract class must be declared with an abstract keyword.

   (b) It can have abstract and non-abstract methods.

(c) It cannot be instantiated.

(d) It can have constructors and static methods also.

(e) It can have final methods which will force the subclass not to change the body of the method.

MIT World Peace University

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 1

Understanding and Implementation of Exception Handling Concepts in C++ and Java.

Practical Report

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

November 15, 2022

Contents

1 Aim and Objectives

Implementation and Understanding of Exception handling in Java and C++, and to learn and use the exception handling mechanisms with try and catch blocks.

2 Problem Statements

2.1 Problem 1 in C++

Define a class Employee consisting following:
Data Members

1. Employee ID
2. Name of Employee
3. Age
4. Income
5. City
6. Vehicle

Member Functions

1. To assign initial values.
2. To display.

Accept Employee ID, Name, Age, Income, City and Vehicle from the user. Create an exception to check the following conditions and throw an exception if the condition does not meet.

- Employee age between 18 and 55
- Employee income between Rs. 50,000 - Rs. 1,00,000 per month
- Employee staying in Pune/ Mumbai/ Bangalore / Chennai
- Employee having 4-wheeler

2.2 Problem 2 in Java

Implement the program to handle the arithmetic exception, ArrayIndexOutofBounds . The user enters the two numbers: n1, n2. The division of n1 and n2 is displayed. If n1, n2 are not integers then program will throw number format exception. If n2 is zero the program will throw Arithmetic exception.

2.3 Problem 3 in Java

Validate the employee record with custom exception Create a class employee with attributes eid, name, age and department. Initialize values through parameterized constructor. If age of employee is not in between 25 and 60 then generate user-defined exception "AgeNotWithinRangeException". If name contains numbers or special symbols raise exception "NameNotValidException". Define the two exception classes.

2.4 Problem 4 in Java

Write a menu-driven program for banking system which accept the personal data for Customer(cid, cname, amount). Implement the user-defined/standard exceptions, wherever required to handle the following situations:

1. Account should be created with minimum amount of 1000 Rs.
2. For withdrawal of amount, if withdrawal Amount is greater than the Amount in the Account.
3. Customer Id should be between 1 and 20 only.
4. Entered amount should be positive.

3 Theory

3.1 Exception Handling

Exception handling is the process of responding to unwanted or unexpected events when a computer program runs. Exception handling deals with these events to avoid the program or system crashing, and without this process, exceptions would disrupt the normal operation of a program.

Exceptions occur for numerous reasons, including invalid user input, code errors, device failure, the loss of a network connection, insufficient memory to run an application, a memory conflict with another program, a program attempting to divide by zero or a user attempting to open files that are unavailable.

When an exception occurs, specialized programming language constructs, interrupt hardware mechanisms or operating system interprocess communication facilities handle the exception.

3.2 Try Throw Catch Block

Exception handling in C++ revolves around these three keywords:
throw- when a program encounters a problem, it throws an exception. The throw keyword helps the program perform the throw.

catch- a program uses an exception handler to catch an exception. It is added to the section of a program where you need to handle the problem. It's done using the catch keyword.

try- the try block identifies the code block for which certain exceptions will be activated. It should be followed by one/more catch blocks. Suppose a code block will raise an exception. The exception

will be caught by a method using try and catch keywords. The try/catch block should surround code that may throw an exception. Such code is known as protected code.

```
  try {
  // the protected code
  } catch( Exception_Name exception1 ) {
  // catch block
  } catch( Exception_Name exception2 ) {
  // catch block
  } catch( Exception_Name exceptionN ) {
  // catch block
  }
```

```
try {
  //  Block of code to try
}
catch(Exception e) {
  //  Block of code to handle errors
}
```

3.3 Catch All

Catch block is used to catch all types of exception. The keyword “catch” is used to catch exceptions. If used like this, it would catch all the exceptions in the try block.

```
  #include <iostream>
  using namespace std;

  void func(int a) {
  try {
    if(a==0) throw 23.33;
    if(a==1) throw 's';
  } catch(...) {
    cout << "Caught Exception!\n";
  }
  }
  int main() {
    func(0);
    func(1);
    return 0;
  }

```

In Java

```
  try {
    //  Block of code to try
  }
  catch(Exception e) {
    //  Block of code to handle errors
  }
```

3.4 Rethrowing Exceptions

If a catch block cannot handle the exception that it was designed to handle, then you can rethrow that exception from that catch block. It causes the original exception to be rethrown.

Because the exception has already been caught at the scope in which the rethrow expression occurs, it is rethrown out to the next dynamically enclosing try block. Therefore, it cannot be handled

by catch blocks at the scope in which the rethrow expression occurred. Any catch blocks for the dynamically enclosing try block have an opportunity to catch the exception.

```
void f() {
  try {
  cout << "In try block of f()" << endl;
  cout << "Throwing exception of type E1" << endl;
  E1 myException;
  throw myException;
  }
  catch (E2& e) {
  cout << "In handler of f(), catch (E2& e)" << endl;
  cout << "Exception: " << e.message << endl;
  throw;
  }
  catch (E1& e) {
  cout << "In handler of f(), catch (E1& e)" << endl;
  cout << "Exception: " << e.message << endl;
  throw;
  }
}
```

4 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers : g++ and gcc on linux for C++, and javac, with JDK 18.0.2 for Java

5 Input

For C++

1. Number of Each Type of Employee
2. Name, Age, Address City, and Salary of Each Employee

For Java

1. The Side of the Square
2. The Radius of the Circle
3. The Length and Breadth of the Rectangle.

6 Output

For C++

1. General Information about Each Employee
2. The Weekly, hourly and commisioned Salary for Respective Employees.

For Java

1. The Area of the Shapes
2. The Location of the Hill Stations
3. The Reason the Hill stations are Famous for.

7 Code

7.1 C++ Implementation of Problem A

```
1
2  // Define a class Employee consisting following:
3  // Data members: a. Employee ID, b. Name of Employee, c. Age, d. Income, e. City,
       f. Vehicle
4  // Member Functions: a. To assign initial values, b. To display.
5
6  // Accept Employee ID, Name, Age, Income, City and Vehicle from the user. Create
7  // an exception to check the following conditions and throw an exception if the
8  // condition does not meet.
9
10 // Employee age between 18 and 55
11 // Employee income between Rs. 50,000 to Rs. 1,00,000 per month
12 // Employee staying in Pune/ Mumbai/ Bangalore / Chennai
13 // Employee having 4-wheeler
14
15 #include <iostream>
16 #include <string.h>
17 using namespace std;
18
19 class Employee
20 {
21 public:
22     int emp_id, age, income;
23     string name, city;
24     bool has_vehicle, data_entered_correctly;
25
26     Employee()
27     {
28         emp_id = 1;
29         age = 30;
30         income = 10000;
31         name = "William";
32         city = "Pune";
33         has_vehicle = true;
34         cout << "The default values are: " << endl;
35         data_entered_correctly = true;
36         display();
37     }
38
39     int accept()
40     {
41
42         cout << "Enter the information of the new Employee : " << endl;
43         cout << "Enter the Employee ID: " << endl;
44         cin >> emp_id;
```

```
45          cout << "Enter the Employee Name: " << endl;
46          cin >> name;
47      age:
48          cout << "Enter the Employee Age: " << endl;
49          try
50          {
51              cin >> age;
52              if (age <= 18 || age >= 55)
53              {
54                  throw age;
55              }
56          }
57          catch (int e)
58          {
59              cout << "Exception Caught!, Age is not in the valid limit. " << endl;
60              data_entered_correctly = false;
61              goto age;
62          }
63      salary:
64          cout << "Enter the Employee basic Salary: " << endl;
65          try
66          {
67              cin >> income;
68              if (income <= 50000 || income >= 100000)
69              {
70                  throw 'a';
71              }
72          }
73          catch (char e)
74          {
75              cout << "Exception Caught!, Income is not in the valid limit. " <<
    endl;
76              data_entered_correctly = false;
77              goto salary;
78          }
79      city:
80          cout << "Enter the Employee Address City: " << endl;
81          try
82          {
83              cin >> city;
84              if (city != "Mumbai" && city != "Pune" && city != "Bangalore")
85              {
86                  throw 1.1;
87              }
88          }
89          catch (double e)
90          {
91              cout << "Exception Caught!, City Entered Incorrectly " << endl;
92              data_entered_correctly = false;
93              goto city;
94          }
95      vehicle:
96          cout << "Does the Employee have a vehicle? (Y, N) " << endl;
97          try
98          {
99              char inp;
100             cin >> inp;
101             if (inp != 'Y')
102             {
```

```
                throw has_vehicle;
            }
        }
        catch (bool e)
        {
            cout << "Exception Caught!, You must have a vehicle!" << endl;
            data_entered_correctly = false;
            goto vehicle;
        }
        data_entered_correctly = true;
        return 0;
    }

    void display()
    {
        if (data_entered_correctly)
        {
            cout << "Employee ID is : " << emp_id << endl;
            cout << "Employee Name: " << name << endl;
            cout << "Employee Age: " << age << endl;
            cout << "Employee Income : " << income << endl;
            cout << "Employee Address City: " << city << endl;
            cout << "Does Employee have a vehicle? : " << has_vehicle << endl;
        }
        else
        {
            cout << "You didnt enter the Data Correctly" << endl;
        }
    }
};

int main()
{
    cout << "Welcome to Assignment 4: Error Safe Employee Data Input Program" <<
    endl;
    Employee obj;
    obj.accept();
    obj.display();
    if (!obj.data_entered_correctly)
    {
        cout << "Please Try again";
    }
    else
    {
        cout << "You have entered the data correctly! " << endl;
    }
    return 0;
}
```

Listing 1: Main.Cpp

7.1.1 C++ Output

```
Welcome to Assignment 4: Error Safe Employee Data Input Program
The default values are:
Employee ID is : 1
Employee Name: William
Employee Age: 30
Employee Income : 10000
```

```
Employee Address City: Pune
Does Employee have a vehicle? : 1
Enter the information of the new Employee :
Enter the Employee ID:
1
Enter the Employee Name:
Mike
Enter the Employee Age:
400
Exception Caught!, Age is not in the valid limit.
Enter the Employee Age:
2000
Exception Caught!, Age is not in the valid limit.
Enter the Employee Age:
40
Enter the Employee basic Salary:
1
Exception Caught!, Income is not in the valid limit.
Enter the Employee basic Salary:
5000
Exception Caught!, Income is not in the valid limit.
Enter the Employee basic Salary:
100000
Exception Caught!, Income is not in the valid limit.
Enter the Employee basic Salary:
60000
Enter the Employee Address City:
Mumbai
Does the Employee have a vehicle? (Y, N)
n
Exception Caught!, You must have a vehicle!
Does the Employee have a vehicle? (Y, N)
Y
Employee ID is : 1
Employee Name: Mike
Employee Age: 40
Employee Income : 60000
Employee Address City: Mumbai
Does Employee have a vehicle? : 1
You have entered the data correctly!
```

Listing 2: Output for Problem 1

7.2 Java Implementation of Problem B

```java
package assignment_4;

import java.util.InputMismatchException;
import java.util.Scanner;

// Implement the program to handle the arithmetic exception, ArrayIndexOutofBounds.
// The user enters the two numbers: n1, n2. The division of n1 and n2 is displayed. If n1, n2
// are not integers then program will throw number format exception. If n2 is zero the
// program will throw Arithmetic exception.

// class NumberFormatException extends Exception
```

```
// {
//     public NumberFormatException(String s)
//     {
//         super(s);
//     }

//     @Override
//     public String getMessage()
//     {
//         return "What is this";
//     }
// }

public class Division {
    int n1;
    int n2;
    int ans;
    Scanner input = new Scanner(System.in);

    void accept() {
        System.out.println("Enter the Numbers that you want to divide. ");
        try {
            n1 = input.nextInt();
            n2 = input.nextInt();
        } catch (InputMismatchException e) {
            System.out.println("Number Format Exception, the format you entered
    does not match. ");
        }
    }

    int divide() {
        try {
            ans = n1 / n2;
        } catch (ArithmeticException e) {
            System.out.println("Exception Caught! Cant divide by Zero!");
        }
        return ans;
    }

}
```

Listing 3: Division.java

7.2.1 Java Output

```

Enter the Employee ID:
1001
Enter the Employee Name
William
Enter the Employee Age:
1
Age not within range
Age is not within the correct range.


Enter the Employee ID:
1
Enter the Employee Name
```

```
William
Enter the Employee Age:
35
Enter the Employee Department:
Sales
Employee ID is : 1
Employee Name: William
Employee Age: 35
Employee Department: Sales
```

Listing 4: Output for Problem 2

7.3 Java Implementation of Problem C

```java
package assignment_4;

import java.util.Scanner;

// Validate the employee record with custom exception
// Create a class employee with attributes eid, name, age and department.
// Initialize values through parameterized constructor. If age of employee is not in between
// 25 and 60 then generate user-defined exception "AgeNotWithinRangeException". If
// name contains numbers or special symbols raise exception "NameNotValidException".
// Define the two exception classes.

class AgeNotWithinRangeException extends Exception {
    public AgeNotWithinRangeException(int s) {
        // super(s);
        System.out.println("Age not within range");
    }

    @Override
    public String getMessage() {
        return "Incorrect Age!";
    }
}

class NameNotValidException extends Exception {
    public NameNotValidException(String s) {
        // super(s);
    }
}

public class Employee {
    Scanner input = new Scanner(System.in);
    int e_id, age;
    String name, department;

    int accept() {
        String specialCharactersString = "1234567890!@#$%&*()'+,-./:;<=>?[]^_`{|}";
        try {
            System.out.println("Enter the Employee ID: ");
            e_id = input.nextInt();
            System.out.println("Enter the Employee Name");
            name = input.next();

```

```
            for (int i = 0; i < name.length(); i++) {
                char ch = name.charAt(i);
                if (specialCharactersString.contains(Character.toString(ch))) {
                    System.out.println(name + " contains special character");
                    throw new NameNotValidException(name);
                }
            }
            System.out.println("Enter the Employee Age: ");
            age = input.nextInt();
            if (age > 60 || age < 25) {
                throw new AgeNotWithinRangeException(age);
            }
            System.out.println("Enter the Employee Department: ");
            department = input.next();

        } catch (AgeNotWithinRangeException e) {
            System.out.println("Age is not within the correct range.");
            return -1;
        } catch (NameNotValidException e) {
            System.out.println("Name not Valid");
            return -1;
        }
        return 0;
    }

    void display() {
        System.out.println("Employee ID is : " + e_id);
        System.out.println("Employee Name: " + name);
        System.out.println("Employee Age: " + age);
        System.out.println("Employee Department: " + department);
    }

}
```

Listing 5: HillStation

7.3.1 Java Output

```

Enter the Employee ID:
1001
Enter the Employee Name
William
Enter the Employee Age:
1
Age not within range
Age is not within the correct range.


Enter the Employee ID:
1
Enter the Employee Name
William
Enter the Employee Age:
35
Enter the Employee Department:
Sales
Employee ID is : 1
Employee Name: William
```

```
Employee Age: 35
Employee Department: Sales
```

Listing 6: Output for Problem 3

7.4 Java Implementation of Problem D

```
package assignment_4;

import java.util.*;

public class Bank {
    public int minimum_sal;
    public int withdrawal_amt;
    public int amount;
    public int c_id;
    Scanner input = new Scanner(System.in);

    public int accept() {
        System.out.println("Enter the customer id: ");
        try {
            c_id = input.nextInt();
            if (c_id > 20 || c_id < 0) {
                throw new Exception("wrong cust id");
            }
        } catch (Exception e) {
            System.out.println("Wrong customer id");
            return 0;
        }

        System.out.println("Enter the Amount in your Account: ");
        try {
            amount = input.nextInt();
            if (amount < 1000) {
                throw new Exception("Amount less than mimimum");
            }
        } catch (Exception e) {
            System.out.println("Mimimum amount cant be less than 1000. ");
            return 0;
        }

        System.out.println("Enter the Withdrawal Amount: ");
        try {
            withdrawal_amt = input.nextInt();
            if (withdrawal_amt > amount) {
                throw new Exception("Withdrawal amount more than amount. ");
            } else {
                amount -= withdrawal_amt;
            }
        } catch (Exception e) {
            System.out.println("Withdrawal Amount more than amount in bank. ");
            return 0;
        }

        return 1;
    }

    public void display() {
        System.out.println("\n\nThe Customer ID is: ");
```

```
        System.out.println(c_id);
        System.out.println("The Amount in the Bank Before Withdrawing was: ");
        System.out.println(amount + withdrawal_amt);
        System.out.println("The Withdrawal Amount is: ");
        System.out.println(withdrawal_amt);
        System.out.println("The Amount Remaining in Bank is: ");
        System.out.println(amount);
    }
}
```

Listing 7: HillStation

```
// P1
// Implement the program to handle the arithmetic exception, ArrayIndexOutofBounds.
// The user enters the two numbers: n1, n2. The division of n1 and n2 is displayed. If n1, n2
// are not integers then program will throw number format exception. If n2 is zero the
// program will throw Arithmetic exception.

// P2
// Validate the employee record with custom exception
// Create a class employee with attributes eid, name, age and department.
// Initialize values through parameterized constructor. If age of employee is not in between
// 25 and 60 then generate user-defined exception "AgeNotWithinRangeException". If
// name contains numbers or special symbols raise exception "NameNotValidException".
// Define the two exception classes.

// P3
// Write a menu-driven program for banking system which accept the personal data for
// Customer(cid, cname, amount).
// Implement the user-defined/standard exceptions, wherever required to handle the
// following situations:
// Account should be created with minimum amount of 1000 rs..
// For withdrawal of amount, if wth_amt greater than amount.
// cid should be in the specific range of 1 to 20.
// Entered amount should be positive.

package assignment_4;

public class Main {

    public static void program_3() {
        Bank obj = new Bank();
        if (obj.accept() == 1) {
            System.out.println("Data entered Correctly!");
            obj.display();
        } else {
            System.out.println("Data entered Incorrectly!");
        }
    }

    public static void program_2() {
        Division d = new Division();
        d.accept();
```

```
        System.out.println("Dividing the inputs: ");
        System.out.println(d.divide());

    }

    public static void program_1() {

        Employee obj = new Employee();
        if (obj.accept() >= 0) {
            obj.display();
        }
    }

    public static void main(String args[]) {

        // program_1();
        program_2();
        program_3();

    }
}
```

Listing 8: HillStation

7.4.1 Java Output

```
Enter the customer id:
1001
Wrong customer id
Data entered Incorrectly!

Enter the customer id:
1
Enter the Amount in your Account:
5000
Enter the Withdrawal Amount:
3000
Data entered Correctly!


The Customer ID is:
1
The Amount in the Bank Before Withdrawing was:
5000
The Withdrawal Amount is:
3000
The Amount Remaining in Bank is:
2000
 krishnaraj@
```

Listing 9: Main.java

8 Conclusion

Thus, learned to use polymorphism and implemented solution of the given problem statement using C++ and Java.

9 FAQs

1. **Why do we use Exception Handling mechanism?**
   Exception handling is the process of responding to unwanted or unexpected events when a computer program runs. Exception handling deals with these events to avoid the program or system crashing, and without this process, exceptions would disrupt the normal operation of a program.

   (a) It helps you avoid the program from crashing

   (b) It helps you set the program control in a more detailed manner

   (c) It helps you manually stop the program safely according to your wish.

```
// Psudo code for exception handling
int a;
try
{
cin>> a;
cout<<33/a; // if a is 0, Exception is thrown, program crashes.
}
catch DivisionByZeroException as e
{
cout<<"cant divide by zero;
}
```

2. **Is it possible to use multiple catch for single throw? Explain?**
   Yes, it is possible to use multiple catch statements for a single throw statement in both C++ and Java. C++ try and catch block works in a specific way, where the throw keyword throws an exception with an integer, or an exception. You can then write multiple catch statements just below the try block.
   In Java, you can throw instances of the Exception class, or throw any of the pre defined exceptions in java.

   Example

```
    int a;
    try
    {
      cin>> a;
      if(a == 0)
      {
        throw a; // int is being thrown.
      }
      else{
        cout<<33/a;
        throw 'a'; // character is being thrown.
      }
    }
    catch (int a)
    {
      cout<<"cant divide by zero;
    }
    catch (char a)
```

```
    {
      cout<<"A is not zero";
    }

```

```
    try {
            withdrawal_amt = input.nextInt();
            if (withdrawal_amt > amount) {
                throw new Exception("Withdrawal amount more than amount. ");
        new_amount = withdrawal_amt / amount;
            } else {
                amount -= withdrawal_amt;
            }
        } catch (Exception e) {
            System.out.println("Withdrawal Amount more than amount in bank. ")
    ;
            return 0;
        } catch (DivisionByZeroException d)
    {
      System.out.println(d);
    }

```

3. **What is Exception Specification?**
   An exception specification is a contract between the function and the rest of the program. It is a guarantee that the function will not throw any exception not listed in its exception specification.

```
    void f1(void) throw(int) {
      printf_s("About to throw 1\n");
      if (1)
         throw 1;
     }

```

```
    void function_name() throw(Exception)
    {
      if (error)
      {
      throw Exception("Error");
      }
    }

```

4. **What is Re-throwing Exception?**
   If a catch block cannot handle the exception that it was designed to handle, then you can rethrow that exception from that catch block. It causes the original exception to be rethrown.

   Because the exception has already been caught at the scope in which the rethrow expression occurs, it is rethrown out to the next dynamically enclosing try block. Therefore, it cannot be handled by catch blocks at the scope in which the rethrow expression occurred. Any catch blocks for the dynamically enclosing try block have an opportunity to catch the exception.

```
    void f() {
      try {
        cout << "In try block of f()" << endl;
        cout << "Throwing exception of type E1" << endl;
```

```
        E1 myException;
        throw myException;
      }
      catch (E2& e) {
        cout << "In handler of f(), catch (E2& e)" << endl;
        cout << "Exception: " << e.message << endl;
        throw;
      }
      catch (E1& e) {
        cout << "In handler of f(), catch (E1& e)" << endl;
        cout << "Exception: " << e.message << endl;
        throw;
      }
    }


```

5. **Explain use of finally keyword in java.**
   Java finally block is a block used to execute important code such as closing the connection, etc.

   Java finally block is always executed whether an exception is handled or not. Therefore, it contains all the necessary statements that need to be printed regardless of the exception occurs or not.

   The finally block follows the try-catch block.

```
class TestFinallyBlock {
  public static void main(String args[]){
  try{
  //below code do not throw any exception
   int data=25/5;
   System.out.println(data);
  }
  //catch won't be executed
  catch(NullPointerException e){
  System.out.println(e);
  }
  //executed regardless of exception occurred or not
   finally {
  System.out.println("finally block is always executed");
  }
```

MIT World Peace University

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 1

Implementation of STL in C++

Practical Report

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

November 15, 2022

Contents

1 Aim and Objectives

- To understand the user of Standard Template Library in C++
- To get familiar with list containers and iterators.

2 Problem Statement

A shop maintains the inventory of items. It stores information of items like ItemCode, ItemName, Quantity and Cost of it in a list of STL. Whenever Customer wants to buy an item, sales person inputs the ItemCode and or ItemName and the system searches in a file and displays whether it is available or not otherwise an appropriate message is displayed. If it is, then the system displays the item details and request for the quantity of items required. If the requested quantity of items are available, the total cost of items is displayed; otherwise the message is displayed as required items not in stock. After purchasing an item, system updates the list. Design a system using a class called Items with suitable data members and member functions. Implement a Menu Driven C++ program using STL concepts.

3 Theory

3.1 Concept of Standard Template Library

The Standard Template Library (STL) is a set of C++ template classes to provide common programming data structures and functions such as lists, stacks, arrays, etc. It is a library of container classes, algorithms, and iterators. It is a generalized library and so, its components are parameterized. It has a lot of useful things that we can use in our own code, without worrying to write lengthy implementations of basic data structure concepts.

STL has 4 components:

1. Algorithms : *They act on containers and provide means for various operations for the contents of the containers.*
2. Containers : *Containers or container classes store objects and data.*
3. Functions :*The STL includes classes that overload the function call operator. Instances of such classes are called function objects or functors.*
4. Iterators : *As the name suggests, iterators are used for working upon a sequence of values. They are the major feature that allows generality in STL.*

3.2 How is STL different from the C++ Standard Library?

The **STL** was written by Alexander Stepanov in the days long before C++ was standardised. The STL was already widely used as a library for C++, giving programmers access to containers, iterators and algorithms. When the standardisation happened, the language committee designed parts of the C++ Standard Library (which is part of the language standard) to very closely match the STL.

Over the years, many people — including prominent book authors, and various websites — have continued to refer to the C++ Standard Library as **The STL** despite the fact that the two entities are separate and that there are some differences. These differences are even more pronounced in the upcoming new C++ standard, which includes various features and significantly alters some classes.

So A lot of the functions and containers were written before the formation of various important libraries that we now use in C++. It is those libraries that are called the "STL". C++ standard libaries are ones written after it using those libraries in part.

3.3 Concept of Containers, Ierators and Algorithms

3.3.1 Containers

In C++, there are generally 3 kinds of STL containers:

- Sequential Containers : *In C++, sequential containers allow us to store elements that can be accessed in sequential order. Internally, sequential containers are implemented as arrays or linked lists data structures.*

  Types of Sequential Containers

  - Array
  - Vector
  - Deque
  - List
  - Forward List

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
  // initialize a vector of int type
  vector<int> numbers = {1, 100, 10, 70, 100};

  // print the vector
  cout << "Numbers are: ";
  for(auto &num: numbers) {
    cout << num << ", ";
  }

  return 0;
}

//Output
Numbers are: 1, 100, 10, 70, 100,
```

- Associative Containers: *In C++, associative containers allow us to store elements in sorted order. The order doesn't depend upon when the element is inserted. Internally, they are implemented as binary tree data structures.* Types of associative Containers.

  - Set
  - Map
  - Multiset
  - Multimap

```
      #include <iostream>
      #include <set>
      using namespace std;

      int main() {

        // initialize a set of int type
        set<int> numbers = {1, 100, 10, 70, 100};

        // print the set
        cout << "Numbers are: ";
        for(auto &num: numbers) {
          cout << num << ", ";
        }

        return 0;
      }
      // Output:
      Numbers are: 1, 10, 70, 100,

```

- Unordered Associative Containers: *In C++, STL Unordered Associative Containers provide the unsorted versions of the associative container. Internally, unordered associative containers are implemented as hash table data structures.* Types of Unordered Associated Containers
  - Unordered Set
  - Unordered Map
  - Unordered Multiset
  - Unordered Multimap

```
#include <iostream>
#include <unordered_set>
using namespace std;

int main() {

  // initialize an unordered_set of int type
  unordered_set<int> numbers = {1, 100, 10, 70, 100};

  // print the set
  cout << "Numbers are: ";
  for(auto &num: numbers) {
    cout << num << ", ";
  }

  return 0;
}
// Output
Numbers are: 70, 10, 100, 1,

```

3.3.2 Iterators

Iterators are one of the four pillars of the Standard Template Library or STL in C++. An iterator is used to point to the memory address of the STL container classes. For better understanding, you can

relate them with a pointer, to some extent. Iterators act as a bridge that connects algorithms to STL containers and allows the modifications of the data present inside the container. They allow you to iterate over the container, access and assign the values, and run different operators over them, to get the desired result.

Applications of Iterators:

1. The primary objective of an iterator is to access the STL container elements and perform certain operations on them.
2. The internal structure of a container does not matter, since the iterators provide common usage for all of them.
3. Iterator algorithms are not dependent on the container type.
4. An iterator can be used to iterate over the container elements. It can also provide access to those elements to modify their values.
5. Iterators follow a generic approach for STL container classes. This way, the programmers dont need to learn about different iterators for different containers.

Example to Demonstrate use of Iterators

```
#include<iostream>
#include<iterator> // for iterators
#include<vector> // for vectors
using namespace std;
int main()
{
  vector<int> ar = { 1, 2, 3, 4, 5 };

  // Declaring iterator to a vector
  vector<int>::iterator ptr;

  // Displaying vector elements using begin() and end()
  cout << "The vector elements are : ";
  for (ptr = ar.begin(); ptr < ar.end(); ptr++)
    cout << *ptr << " ";

  return 0;
}
//Output
The vector elements are : 1 2 3 4 5
```

3.3.3 Algorithms

STL provide different types of algorithms that can be implemented upon any of the container with the help of iterators. Thus now we don't have to define complex algorithm instead we just use the built in functions provided by the algorithm library in STL.

Algorithm functions provided by algorithm library works on the iterators, not on the containers. Thus one algorithm function can be used on any type of container. Use of algorithms from STL saves time, effort, code and are very reliable.

There are many types of algorithms already implemented reliably in the STL. Here we will use a simple Non Modifying Algorithm as an example.

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
int main ()
{
  int values[] = {5,1,6,9,10,1,12,5,5,5,1,8,9,7,46};
  int count_5 = count(values, values+15, 5);
  cout<<"The number of times '5' appears in array= "<<count_5;
  return 0;
}
// Output
The number of times 5 appears in an array= 4
```

4 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers : g++ and gcc on linux for C++

5 Input

For C++

1. Basic menu to add new elements, or purchase an item.
2. The Details of the item to add.
3. The quantity and code of the product that you wanna purchase.

6 Output

For C++

1. A list of all the elements in the Database.
2. Their codes and Quantities
3. Appropriate messages after each action is performed.

7 Code

7.1 C++ Implementation of Problem A

```cpp
// Shop has item code, item name, quantity, and the cost.
// Input would be some item name, where we would first add things to the shop
    inventory
// Another option would be to check for the item in the database which is a list.
// If the item is found in the shop then you are supposed to ask them the number
    of items.
// if its available then you sell it, or else you tell them that that may items
    arent in stock.
```

```
6
7  #include <iostream>
8  #include <list>
9  #include <iomanip>
10 using namespace std;
11
12 // struct so we can put it in a single linked list.
13 struct Items
14 {
15     int item_code;
16     string item_name;
17     int item_quantity;
18     int item_cost;
19     Items(int a, string b, int q, int c) : item_code(a), item_name(b),
       item_quantity(q), item_cost(c)
20     {
21     }
22 } currentItem(0, "", 0, 0), itemToAdd(0, "", 0, 0);
23
24 // linked list items so we can put objects of structures in it easily.
25 list<Items> items = {
26     Items(101, "Burger", 10, 200),
27     Items(102, "Fries", 10, 129),
28     Items(103, "Ice Cream", 10, 40),
29     Items(104, "Coke", 10, 50),
30 };
31
32 bool itemFound = false;
33 // returns true or false depending on whether the item was found
34 bool searchItem(int itemCode)
35 {
36     int i = 0;
37     for (list<Items>::iterator it = items.begin(); it != items.end(); it++, i++)
38     {
39         struct Items temp = *it;
40         if (temp.item_code == itemCode)
41         {
42             currentItem = temp;
43             return true;
44         }
45     }
46     return false;
47 }
48
49 // inserts items in the list
50 void insertItems()
51 {
52     cout << "Enter the details of the Item that you wanna enter to the database"
       << endl;
53 i:
54     cout << "Enter the Code of the new Item: ";
55     cin >> itemToAdd.item_code;
56     if (searchItem(itemToAdd.item_code))
57     {
58         cout << "Item already exists, try another code!" << endl;
59         goto i;
60     }
61     cout << "Enter the Name of the new Item: ";
62     cin >> itemToAdd.item_name;
```

```
63
64      cout << "Enter the Quantity of the new Item: ";
65      cin >> itemToAdd.item_quantity;
66
67      cout << "Enter the Cost of the new Item: ";
68      cin >> itemToAdd.item_cost;
69
70      items.push_back(itemToAdd);
71      cout << "Item added successfully" << endl;
72  }
73
74  // just find the element at the element currentItemIndex, and replace it with the
        currentItem struct object.
75  void updateItems(int updatedItemcost, int updatedQuantity, int currentItemCode)
76  {
77      for (list<Items>::iterator it = items.begin(); it != items.end(); it++)
78      {
79          struct Items temp = *it;
80          if (temp.item_code == currentItemCode)
81          {
82              currentItem.item_quantity = updatedQuantity;
83              currentItem.item_cost = updatedItemcost;
84              items.erase(it);
85              items.push_back(currentItem);
86          }
87      }
88  }
89
90  // just displays the items in a table nicely
91  void displayItems()
92  {
93      struct Items tempItem(0, "", 0, 0);
94      cout << "The Items that you can buy are: " << endl;
95
96      cout << setw(20) << "ITEM CODE" << setw(20) << "ITEM NAME" << setw(20) << "
        ITEM QUANTITY" << setw(20) << "ITEM COST" << endl;
97      for (list<Items>::iterator it = items.begin(); it != items.end(); it++)
98      {
99          tempItem = *it;
100         cout << setw(20) << tempItem.item_code << setw(20) << tempItem.item_name
        << setw(20) << tempItem.item_quantity << setw(20) << tempItem.item_cost << endl
        ;
101     }
102 }
103
104 // display the things, and check if the selected item code by the user exists in
        the thing by calling searchItem. Then input the item quantity, check for it,
        and update the currentItem object.
105 // then call the updateItem function.
106 bool PurchaseItems()
107 {
108     int selectedItemCode = -1;
109     int selectedQuantity = 0;
110 c:
111     try
112     {
113         cout << "Please enter the code of the item that you wanna buy" << endl;
114         cin >> selectedItemCode;
115         if (!searchItem(selectedItemCode))
```

```
        {
            throw selectedItemCode;
        }
    l:
        cout << "Enter the Quantity of the Item that you wanna buy. "
             << endl;
        cout << "Max quantity is: " << currentItem.item_quantity << endl;
        cin >> selectedQuantity;
        try
        {
            if (selectedQuantity > currentItem.item_quantity || selectedQuantity
    <= 0)
            {
                throw selectedQuantity;
            }
            else
            {
                cout << "That will be: " << currentItem.item_cost *
    selectedQuantity << " Rupees" << endl;
                cout << "Thank you for Purchasing!" << endl;
                updateItems(currentItem.item_cost, currentItem.item_quantity -
    selectedQuantity, selectedItemCode);
            }
        }
        catch (int something)
        {
            cout << "Quantity you entered is not sensible! Try again!" << endl;
            goto l;
        }
        return true;
    }
    catch (int something)
    {
        cout << "The code of the item that you entered doesnt exist. Try again. "
    << endl;
        goto c;
    }
    return true;
}

// function to change the price of the item.
bool changePrice()
{
    int selectedItemCode = -1;
    int updatedCost = 0;
c:
    try
    {
        cout << "Please enter the code of the item that you wanna Change the Price
     of" << endl;
        cin >> selectedItemCode;
        if (!searchItem(selectedItemCode))
        {
            throw selectedItemCode;
        }
    l:
        cout << "Enter the Updated Price of the Item."
             << endl;
        cout << "Current Price is: " << currentItem.item_cost << endl;
```

```
170         cin >> updatedCost;
171         try
172         {
173             if (updatedCost <= 0)
174             {
175                 throw updatedCost;
176             }
177             else
178             {
179                 cout << "Done!" << endl;
180                 updateItems(updatedCost, currentItem.item_quantity,
    selectedItemCode);
181             }
182         }
183         catch (int something)
184         {
185             cout << "Cost you entered is not sensible! Try again!" << endl;
186             goto l;
187         }
188         return true;
189     }
190     catch (int something)
191     {
192         cout << "The code of the item that you entered doesnt exist. Try again. "
    << endl;
193         goto c;
194     }
195     return true;
196 }
197
198 // function to change the quantity of the item.
199 bool changeQuantity()
200 {
201     int selectedItemCode = -1;
202     int updatedQuantity = 0;
203 c:
204     try
205     {
206         cout << "Please enter the code of the item that you wanna update" << endl;
207         cin >> selectedItemCode;
208         if (!searchItem(selectedItemCode))
209         {
210             throw selectedItemCode;
211         }
212     l:
213         cout << "Enter the updated Quantity"
214              << endl;
215         cout << "Current quantity is: " << currentItem.item_quantity << endl;
216         cin >> updatedQuantity;
217         try
218         {
219             if (updatedQuantity <= 0)
220             {
221                 throw updatedQuantity;
222             }
223             else
224             {
225                 cout << "Done!" << endl;
226                 updateItems(currentItem.item_cost, updatedQuantity,
```

```
    selectedItemCode);
227             }
228         }
229         catch (int something)
230         {
231             cout << "Quantity you entered is not sensible! Try again!" << endl;
232             goto l;
233         }
234         return true;
235     }
236     catch (int something)
237     {
238         cout << "The code of the item that you entered doesnt exist. Try again. "
    << endl;
239         goto c;
240     }
241     return true;
242 }
243
244 int main()
245 {
246     int choice = 0;
247     struct Items;
248     cout << "Welcome to McRonalds" << endl;
249
250     do
251     {
252         cout << endl
253              << "What do you wanna do?\n\
254         1. Add new Items\n\
255         2. Purchase Item\n\
256         3. Change Price of Item\n\
257         4. Change Quantity of Item\n\
258         5. View Items\n\
259         6. Quit\n"
260              << endl;
261         cin >> choice;
262         switch (choice)
263         {
264         case 1:
265             insertItems();
266             break;
267         case 2:
268             displayItems();
269             PurchaseItems();
270             break;
271         case 3:
272             displayItems();
273             changePrice();
274             break;
275         case 4:
276             displayItems();
277             changeQuantity();
278             break;
279         case 5:
280             displayItems();
281             break;
282         case 6:
283             cout << "Thanks for Visiting our store!" << endl;
```

```
            break;
        default:
            cout << "Sorry, we cant do that in this store" << endl;
            break;
        }
        itemFound = false;
    } while (choice != 6);
    return 0;
}
```

Listing 1: Main.Cpp

7.1.1 C++ Input and Output

```
Welcome to McRonalds

What do you wanna do?
        1. Add new Items
        2. Purchase Item
        3. Change Price of Item
        4. Change Quantity of Item
        5. View Items
        6. Quit

5
The Items that you can buy are:
           ITEM CODE           ITEM NAME       ITEM QUANTITY           ITEM COST
                 101              Burger                  10                 200
                 102               Fries                  10                 129
                 103           Ice Cream                  10                  40
                 104                Coke                  10                  50

What do you wanna do?
        1. Add new Items
        2. Purchase Item
        3. Change Price of Item
        4. Change Quantity of Item
        5. View Items
        6. Quit

1
Enter the details of the Item that you wanna enter to the database
Enter the Code of the new Item: 101
Item already exists, try another code!
Enter the Code of the new Item: 108
Enter the Name of the new Item: oreo
Enter the Quantity of the new Item: 50
Enter the Cost of the new Item: 40
Item added successfully

What do you wanna do?
        1. Add new Items
        2. Purchase Item
        3. Change Price of Item
        4. Change Quantity of Item
        5. View Items
        6. Quit

5
```

```
The Items that you can buy are:
           ITEM CODE           ITEM NAME       ITEM QUANTITY           ITEM COST
                 101              Burger                  10                 200
                 102               Fries                  10                 129
                 103           Ice Cream                  10                  40
                 104                Coke                  10                  50
                 108                oreo                  50                  40

What do you wanna do?
        1. Add new Items
        2. Purchase Item
        3. Change Price of Item
        4. Change Quantity of Item
        5. View Items
        6. Quit

2
The Items that you can buy are:
           ITEM CODE           ITEM NAME       ITEM QUANTITY           ITEM COST
                 101              Burger                  10                 200
                 102               Fries                  10                 129
                 103           Ice Cream                  10                  40
                 104                Coke                  10                  50
                 108                oreo                  50                  40
Please enter the code of the item that you wanna buy
108
Enter the Quantity of the Item that you wanna buy.
Max quantity is: 50
55
Quantity you entered is not sensible! Try again!
Enter the Quantity of the Item that you wanna buy.
Max quantity is: 50
4
That will be: 160 Rupees
Thank you for Purchasing!

What do you wanna do?
        1. Add new Items
        2. Purchase Item
        3. Change Price of Item
        4. Change Quantity of Item
        5. View Items
        6. Quit

3
The Items that you can buy are:
           ITEM CODE           ITEM NAME       ITEM QUANTITY           ITEM COST
                 101              Burger                  10                 200
                 102               Fries                  10                 129
                 103           Ice Cream                  10                  40
                 104                Coke                  10                  50
                 108                oreo                  46                  40
Please enter the code of the item that you wanna Change the Price of
104
Enter the Updated Price of the Item.
Current Price is: 50
45
Done!
```

```
What do you wanna do?
        1. Add new Items
        2. Purchase Item
        3. Change Price of Item
        4. Change Quantity of Item
        5. View Items
        6. Quit

4
The Items that you can buy are:
           ITEM CODE           ITEM NAME       ITEM QUANTITY           ITEM COST
                 101              Burger                  10                 200
                 102               Fries                  10                 129
                 103           Ice Cream                  10                  40
                 108                oreo                  46                  40
                 104                Coke                  10                  45
Please enter the code of the item that you wanna update
108
Enter the updated Quantity
Current quantity is: 46
50
Done!

What do you wanna do?
        1. Add new Items
        2. Purchase Item
        3. Change Price of Item
        4. Change Quantity of Item
        5. View Items
        6. Quit

5
The Items that you can buy are:
           ITEM CODE           ITEM NAME       ITEM QUANTITY           ITEM COST
                 101              Burger                  10                 200
                 102               Fries                  10                 129
                 103           Ice Cream                  10                  40
                 104                Coke                  10                  45
                 108                oreo                  50                  40

What do you wanna do?
        1. Add new Items
        2. Purchase Item
        3. Change Price of Item
        4. Change Quantity of Item
        5. View Items
        6. Quit

6
Thanks for Visiting our store!
```

Listing 2: Output for Problem 1

8 Conclusion

Thus, the purpose of the STL libraries in C++ was understood, and implemented successfully. Containers like lists and iterators for them were also used and understood.

9 FAQs

1. **What are class templates? How are they created? What is the need for class templates?**
   Templates are powerful features of C++ which allows us to write generic programs. There are two ways we can implement templates:

   - Function Templates
   - Class Templates

   Similar to function templates, we can use class templates to create a single class to work with different data types. There are few, but important reasons to use class templates:

   - Class templates come in handy as they can make our code shorter and more manageable.
   - If you have various functions that you wanna implement on a set of data, but you arent sure which data type will be given as input, then you can use class templates.
   - Example: If you creating a calculator, a class template to include basic functions like addition, subtraction etc would be perfect as you can get an integer or a floating point value as your input for your calculator.

2. **Create a template for bubble sort functions.**

```
template <class T>
T bubbleSort(T arr[], int n)
{
  int i, j;
  for (i = 0; i < n - 1; i++)
    for (j = 0; j < n - i - 1; j++)
      if (arr[j] > arr[j + 1])
        swap(arr[j], arr[j + 1]);
}


```

3. **Explain with example, how Function Templates are implemented?**

```
#include <iostream>
using namespace std;
template<class T> T add(T &a,T &b)
{
  T result = a+b;
  return result;
}
int main()
{
  int i =2;
  int j =3;
  float m = 2.3;
  float n = 1.2;
  cout<<"Addition of i and j is :"<<add(i,j);
  cout<<'\n';
  cout<<"Addition of m and n is :"<<add(m,n);
  return 0;
```

```
}

```

4. **Explain with example how can a class template be created.**

```
// Class template
template <class T>
class Number {
  private:
  // Variable of type Tf
  T num;

  public:
  Number(T n) : num(n) {}   // constructor

  T getNum() {
    return num;
  }
};

int main() {

  // create object with int type
  Number<int> numberInt(7);

  // create object with double type
  Number<double> numberDouble(7.7);

  cout << "int Number = " << numberInt.getNum() << endl;
  cout << "double Number = " << numberDouble.getNum() << endl;

  return 0;
}
// Output
int Number = 7
double Number = 7.7

```

5. **Explain Generic functions and Generic class.**

- Generic functions use the concept of a function template. Generic functions define a set of operations that can be applied to the various types of data.
- The type of the data that the function will operate on depends on the type of the data passed as a parameter.
- For example, Quick sorting algorithm is implemented using a generic function, it can be implemented to an array of integers or array of floats.
- A Generic function is created by using the keyword template. The template defines what function will do.
- Just like a class is a collection of a bunch of functions, that can be inherited and instantiated at once, generic functions are similar in that manner, except in a generic class, you could use a generic data type, and this would be applicable and useful for implementing each function in the Class. So each function in the class can be called and can manipulate variables of the genereic data type for which the class is defined.

MIT World Peace University

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 1

Collection Frameworks in Java

Practical Report
Assignment 6

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

November 25, 2022

Contents

1 Aim and Objectives

Aim

Demonstrating the use of Collection Frameworks in Java.

Objective

- To study the concept of collection framework
- To get familiar with features of collection framework in Java

2 Problem Statement

Write a Java Program to:

- Create a new array list and print out the collection by position
- Add some elements (string)
- Search an element in an array list
- Reverse elements in an array list
- Test an array list is empty or not
- Join two array lists

3 Theory

3.1 Concept of collection framework in Java

The Collection in Java is a framework that provides an architecture to store and manipulate the group of objects.

Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

Here is an Example using the ArrayList interface.

```
import java.util.*;
class TestJavaCollection1{
  public static void main(String args[]){
    ArrayList<String> list=new ArrayList<String>();//Creating arraylist
    list.add("Ravi");//Adding object in arraylist
    list.add("Vijay");
    list.add("Ravi");
    list.add("Ajay");
    //Traversing list through Iterator
    Iterator itr=list.iterator();
    while(itr.hasNext()){
      System.out.println(itr.next());
    }
  }
}
```

3.2 Benefit of Generics in Collections Framework

The Java Generics programming is introduced in J2SE 5 to deal with type-safe objects. It makes the code stable by detecting the bugs at compile time.

Before generics, we can store any type of objects in the collection, i.e., non-generic. Now generics force the java programmer to store a specific type of objects.

The Advantages are:

- Type-safety : We can hold only a single type of objects in generics. It doesn?t allow to store other objects.
- Type casting is not required: There is no need to typecast the object.
- Compile-time checking: It is checked at compile time so problem will not occur at runtime. It is far better to handle the problem at compile time than runtime.

3.3 Basic interfaces of Java Collections Framework

Java Collection means a single unit of objects. Java Collection framework provides many basic interfaces.

- ArrayList: It is used to create a dynamic array. It is similar to vector.
- LinkedList: It is used to create a linked list. It is the implementation of the List and Deque interfaces.
- Vector: It is similar to ArrayList. It is also used to create a dynamic array.
- HashSet: It is used to create a collection that uses a hash table for storage.
- TreeSet: It is used to create a collection that uses a tree for storage.
- HashMap: It is used to create a map that uses a hash table for storage.
- TreeMap: It is used to create a map that uses a tree for storage.

3.4 Types of collections allowed by the Java collections framework, Explain with suitable syntax and examples

4 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers : g++ and gcc on linux for C++, and javac, with JDK 18.0.2 for Java

5 Input

For C++

1. Number of Each Type of Employee
2. Name, Age, Address City, and Salary of Each Employee

For Java

1. The Side of the Square
2. The Radius of the Circle
3. The Length and Breadth of the Rectangle.

6 Output

For C++

1. General Information about Each Employee
2. The Weekly, hourly and commisioned Salary for Respective Employees.

For Java

1. The Area of the Shapes
2. The Location of the Hill Stations
3. The Reason the Hill stations are Famous for.

7 Code

```java
// Write a Java program to
// create a new array list and print out the collection by position
// add some elements (string)
// search an element in an array list
// reverse elements in an array list
// test an array list is empty or not
// join two array lists

// Krishnaraj Thadesar
// PA20
// Batch A1
// Assignment 6 in OOPCJ

import java.util.*;

import javax.lang.model.util.ElementScanner14;

import java.lang.*;

public class assignment_6 {
    static Scanner input = new Scanner(System.in);
    static ArrayList<String> arrayList, list1, list2;
    static String list1_input, list2_input, main_list_input;

    public static void main(String[] args) {
        System.out.println("Welcome to Assignment 6! What do you want to do? ");
        int choice = 0;
        arrayList = new ArrayList<String>();
        while (choice != 7) {
```

```
30              System.out.println(
31                      "\n\n1. Add some elements (string)\n2. Search an element in an
     array list\n3. Reverse elements in an array list\n4. Test an array list is
    empty or not\n5. Join two array lists\n6. Display All Elements\n7. Quit\n");
32              choice = input.nextInt();
33              switch (choice) {
34                  case 1:
35                      System.out.println("Enter the Element that you want to enter:
    ");
36                      main_list_input = input.next();
37                      arrayList.add(main_list_input);
38                      break;
39                  case 2:
40                      System.out.println("Enter the Element that you want to Search:
     ");
41                      main_list_input = input.next();
42                      // search an element in ArrayList
43                      if (arrayList.contains(main_list_input)) {
44                          System.out.println("Element found");
45                      } else {
46                          System.out.println("Element not found");
47                      }
48                      break;
49                  case 4:
50                      System.out.println("Checking if Arraylist is Empty: ");
51                      if (arrayList.isEmpty()) {
52                          System.out.println("Array list is empty!");
53                      } else {
54                          System.out.println("Array list isnt empty!");
55                      }
56                      break;
57                  case 5:
58                      System.out.println("Joining 2 Array lists, so enter them: ");
59                      System.out.println("Enter List 1, enter \"done\" to stop
    entering");
60                      list1 = new ArrayList<String>();
61                      list1_input = new String();
62                      while (!list1_input.equals("done")) {
63                          list1_input = input.next();
64                          if (!list1_input.equals("done")) {
65                              list1.add(list1_input);
66                          }
67                      }
68                      System.out.println("Enter List 2, enter \"done\" to stop
    entering");
69                      list2 = new ArrayList<String>();
70                      list2_input = new String();
71                      while (!list2_input.equals("done")) {
72                          list2_input = input.next();
73                          if (!list2_input.equals("done")) {
74                              list2.add(list2_input);
75                          }
76                      }
77                      // merge 2 Arraylists
78                      list1.addAll(list2);
79                      System.out.println("Displaying all the elements");
80                      for (String iterable_element : list1) {
81                          System.out.println(iterable_element);
82                      }
```

```
                    break;
                case 6:
                    System.out.println("Displaying all the elements");
                    for (String iterable_element : arrayList) {
                        System.out.println(iterable_element);
                    }
                    break;
                case 3:
                    System.out.println("Reversing the Elements of the List");
                    Collections.reverse(arrayList);
                    System.out.println("Displaying all the elements");
                    for (String iterable_element : arrayList) {
                        System.out.println(iterable_element);
                    }
                    break;
                case 7:
                    System.out.println("Quitting!");
                    System.exit(0);
                default:
                    break;
            }
        }
    }
}
```

Listing 1: Output

7.0.1 Output

```

Welcome to Assignment 6! What do you want to do?


1. Add some elements (string)
2. Search an element in an array list
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit

1
Enter the Element that you want to enter:
a


1. Add some elements (string)
2. Search an element in an array list
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit

1
Enter the Element that you want to enter:
b


```

```
1. Add some elements (string)
2. Search an element in an array list
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit

1
Enter the Element that you want to enter:
c


1. Add some elements (string)
2. Search an element in an array list
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit

1
Enter the Element that you want to enter:
4


1. Add some elements (string)
2. Search an element in an array list
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit

2
Enter the Element that you want to Search:
a
Element found


1. Add some elements (string)
2. Search an element in an array list
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit

3
Reversing the Elements of the List
Displaying all the elements
4
c
b
a


1. Add some elements (string)
2. Search an element in an array list
```

```
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit

4
Checking if Arraylist is Empty:
Array list isnt empty!


1. Add some elements (string)
2. Search an element in an array list
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit

5
Joining 2 Array lists, so enter them:
Enter List 1, enter "done" to stop entering
1
2
3
done
Enter List 2, enter "done" to stop entering
33
4
55
done
Displaying all the elements
1
2
3
33
4
55


1. Add some elements (string)
2. Search an element in an array list
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit


6
Displaying all the elements
4
c
b
a


1. Add some elements (string)
2. Search an element in an array list
```

```
3. Reverse elements in an array list
4. Test an array list is empty or not
5. Join two array lists
6. Display All Elements
7. Quit

7
Quitting!
```

Listing 2: Output

8 Conclusion

Thus, learnt the use of collection framework in java and performed array list operations

9 FAQs

1. **Why do we use collection framework?**
   Java Collection Framework offers the capability to Java Collection to represent a group of elements in classes and Interfaces.

   Java Collection Framework enables the user to perform various data manipulation operations like storing data, searching, sorting, insertion, deletion, and updating of data on the group of elements. Followed by the Java Collections Framework, you must learn and understand the Hierarchy of Java collections and various descendants or classes and interfaces involved in the Java Collections.

   - When input size is dynamic.
   - Whenever we are required to store heterogeneous data.
   - When data grows and shrinks frequently.

2. **Which is best collection framework in Java?**
   There are several very important collection frameworks, in Java. They are:

   - ArrayList
   - LinkedList
   - Vector
   - HashSet
   - TreeSet
   - HashMap
   - TreeMap

   All of them are important, and useful in their own way. So we cannot say one of them is the best one, as it depends on the requirements.

3. **What is difference between array and collection?**

   - Arrays are fixed in size, whereas collections are dynamic in size.
   - Arrays are homogeneous, whereas collections are heterogeneous.
   - Arrays are not type safe, whereas collections are type safe.
   - Arrays are not thread safe, whereas collections are thread safe.
   - Arrays are not synchronized, whereas collections are synchronized.

4. **What is HashMap in Java?**
   HashMap is a part of the Java Collections Framework. It is a class that implements the Map interface. It is used to store key-value pairs. It is similar to Hashtable, but it is not synchronized. It allows one null key and multiple null values. It maintains no order.

   It provides the functionality of the hash table data structure. It stores elements in key/value pairs. Here, keys are unique identifiers used to associate each value on a map.

MIT World Peace University

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 1

Multithreading using Thread Class and Runnable Interface in Java

---

Practical Report
Assignment 7

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

November 23, 2022

Contents

1 Aim and Objectives

Aim

Implementing Solutions on Multithreading using Thread Class and Runnable Interface

Objectives

1. To understand Multithreading in Java
2. To learn two different ways to create threads in Java

2 Problem Statements

2.1 Problem 1 in Java

Write a program to create a multithreaded calculator that does addition, subtraction, multiplication, and division using separate threads. Additionally also handle '/ by zero' exception by the division method.

2.2 Problem 2 in Java

Print even and odd numbers in increasing order using two threads in Java

3 Theory

3.1 Multithreading in Java

In Java, Multithreading refers to a process of executing two or more threads simultaneously for maximum utilization of the CPU. A thread in Java is a lightweight process requiring fewer resources to create and share the process resources.Multithreading and Multiprocessing are used for multitasking in Java, but we prefer multithreading over multiprocessing.

3.2 Life Cycle of a Thread in Java

There are five states a thread has to go through in its life cycle. This life cycle is controlled by JVM (Java Virtual Machine). These states are:

1. New : In this state, a new thread begins its life cycle. This is also called a born thread. The thread is in the new state if you create an instance of Thread class but before the invocation of the start() method.
2. Runnable : A thread becomes runnable after a newly born thread is started. In this state, a thread would be executing its task.
3. Running : When the thread scheduler selects the thread then, that thread would be in a running state.
4. Non-Runnable (Blocked) : The thread is still alive in this state, but currently, it is not eligible to run.
5. Terminated : A thread that is in a terminated state does not consume ant cycle of the CPU.

3.3 Ways of Creating a Thread

There are multiple ways of creating Threads in Java

1. By Extending the Thread Class

```
class Multi extends Thread{
  public void run(){
    System.out.println("thread is running...");
  }
  public static void main(String args[]){
    Multi t1=new Multi();
    t1.start();
  }
}

```

2. By Implementing the Runnable Interface

```
class Multi3 implements Runnable{
  public void run(){
    System.out.println("thread is running...");
  }

  public static void main(String args[]){
    Multi3 m1=new Multi3();
    Thread t1 =new Thread(m1);   // Using the constructor Thread(Runnable r)
    t1.start();
  }
}

```

3. Using the Thread Class

```
public class MyThread1
{
  // Main method
  public static void main(String argvs[])
  {
    // creating an object of the Thread class using the constructor Thread(
    String name)
    Thread t= new Thread("My first thread");

    // the start() method moves the thread to the active state
    t.start();
    // getting the thread name by invoking the getName() method
    String str = t.getName();
    System.out.println(str);
  }
}
```

3.4 Performing Multiple tasks by multiple threads

You could just create multiple classes that perform their own tasks, each a child of the Thread class. So you could then invoke instances of those classes, and run them as multiple threads.

```
class MultithreadEx3 extends Thread
{
```

```
    public void run()
    {
        System.out.println("Start task one");
    }
}
class MultithreadEx4 extends Thread
{
    public void run()
    {
        System.out.println("Start task two");
    }
}
class Run
{
    public static void main(String args[])
    {
        MultithreadEx3 th1 = new MultithreadEx3();
        MultithreadEx4 th2 = new MultithreadEx4();
        th1.start();
        th2.start();
    }
}
```

3.5 Thread Scheduler

A component of Java that decides which thread to run or execute and which thread to wait is called a thread scheduler in Java. In Java, a thread is only chosen by a thread scheduler if it is in the runnable state.

However, if there is more than one thread in the runnable state, it is up to the thread scheduler to pick one of the threads and ignore the other ones. There are some criteria that decide which thread will execute first. There are two factors for scheduling a thread i.e. Priority and Time of arrival.

- *Priority*: Priority of each thread lies between 1 to 10. If a thread has a higher priority, it means that thread has got a better chance of getting picked up by the thread scheduler.
- *Time of Arrival*: Suppose two threads of the same priority enter the runnable state, then priority cannot be the factor to pick a thread from these two threads. In such a case, arrival time of thread is considered by the thread scheduler. A thread that arrived first gets the preference over the other threads.

3.6 Joining a Thread in Java

java.lang.Thread class provides the join() method which allows one thread to wait until another thread completes its execution.

If t is a Thread object whose thread is currently executing, then t.join() will make sure that t is terminated before the next instruction is executed by the program. If there are multiple threads calling the join() methods that means overloading on join allows the programmer to specify a waiting period.

However, as with sleep, join is dependent on the OS for timing, so you should not assume that join will wait exactly as long as you specify. There are three overloaded join functions.

```java
public class JoinExample1 extends Thread
{
    public void run()
    {
        for(int i=1; i<=4; i++)
        {
            try
            {
                Thread.sleep(500);
            }catch(Exception e){System.out.println(e);}
            System.out.println(i);
        }
    }
    public static void main(String args[])
    {
        JoinExample1 thread1 = new JoinExample1();
        JoinExample1 thread2 = new JoinExample1();
        JoinExample1 thread3 = new JoinExample1();
        thread1.start();
       try
        {
        thread1.join();
        }catch(Exception e){System.out.println(e);}
        thread2.start();
        thread3.start();
    }
}
```

4 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers : g++ and gcc on linux for C++, and javac, with JDK 18.0.2 for Java

5 Input

For Problem 1

1. 2 numbers
2. Choice about what to do with those numbers

For Problem 2

1. The Maximum limit up to which the user wants to see the odd and even numbers printed

6 Output

For Problem 1

1. Menu about what to do with numbers
2. Output of the calculation done with those numbers

For Problem 2

1. Even numebers and Odd numbers in Ascending order upto the specified limit.

7 Code

7.1 Java Implementation of Problem 1

```java
// Krishnaraj Thadesar
// Batch A1, PA20
// OOPCJ Assignment 7.1
// Write a program to create a multithreaded calculator that does addition, subtraction,
// multiplication, and division using separate threads.
// Additionally also handle '/ by zero' exception by the division method.


import java.lang.Thread;
import java.util.Scanner;

class Calculator extends Thread implements Runnable {
    public int a, b, what_to_do = 0;

    Calculator(int a, int b, int choice, String name) {
        this.a = a;
        this.b = b;
        this.what_to_do = choice;
        this.setName(name);
    }

    @Override
    public synchronized void start() {
        System.out.println("Starting the Thread");
        System.out.println("The Name of this Thread is: " + getName());
        super.start();
    }

    @Override
    public void run() {
        switch (what_to_do) {
            case 1:
                System.out.println(a + b);
                break;
            case 2:
                System.out.println(a - b);
                break;
            case 3:
                System.out.println(a * b);
                break;
            case 4:
                try {
                    System.out.println(a / b);
                } catch (ArithmeticException e) {
                    System.out.println("You cant Divide by Zero!");
                }
                break;
            default:
```

```
                break;
            }
        }
}

public class assignment_7_problem_1 {
    public static Calculator add, sub, mul, div;
    public static Scanner input = new Scanner(System.in);

    public static void main(String[] args) {
        int choice = 0;
        int a, b;
        System.out.println("Welcome To Thread Calculator of Assignment 7");
        while (choice != 5) {
            System.out.println("What would you like to do? ");
            System.out.println(
                    "1. Addition of 2 Numbers\n2. Subtraction of 2 Numbers\n3. Multiplication of 2 Numbers\n4. Division of 2 Numbers\n\n");
            choice = input.nextInt();
            if (choice == 5) {
                break;
            }
            System.out.println("Enter the 2 Numbers");
            a = input.nextInt();
            b = input.nextInt();
            switch (choice) {
                case 1:
                    System.out.println("You have chosen Addition!");
                    add = new Calculator(a, b, choice, "Adder");
                    try {
                        add.start();
                        add.join();
                    } catch (Exception e) {
                        System.out.println("Got some problem with making the thread!");
                        System.out.println(e);
                    }
                    break;
                case 2:
                    System.out.println("You have chosen Subtraction!");
                    sub = new Calculator(a, b, choice, "Subtractor");
                    try {
                        sub.start();
                        sub.join();
                    } catch (Exception e) {
                        System.out.println("Got some problem with making the thread!");
                        System.out.println(e);
                    }
                    break;
                case 3:
                    System.out.println("You have chosen Multiplication!");
                    mul = new Calculator(a, b, choice, "Multiplier");
                    try {
                        mul.start();
                        mul.join();
                    } catch (Exception e) {
                        System.out.println("Got some problem with making the thread!");
```

```
                        System.out.println(e);
                    }
                    break;
                case 4:
                    System.out.println("You have chosen Division!");
                    div = new Calculator(a, b, choice, "Divider");
                    try {
                        div.start();
                        div.join();
                    } catch (Exception e) {
                        System.out.println("Got some problem with making the
    thread!");
                        System.out.println(e);
                    }
                    break;
                case 5:
                    System.out.println("You have chosed to Exit!");
                default:
                    break;
            }

        }
        System.exit(0);
    }
}
```

Listing 1: Probelm 1.java

7.1.1 Java Output

```
Welcome To Thread Calculator of Assignment 7
What would you like to do?
1. Addition of 2 Numbers
2. Subtraction of 2 Numbers
3. Multiplication of 2 Numbers
4. Division of 2 Numbers


1
Enter the 2 Numbers
2
2
You have chosen Addition!
Starting the Thread
The Name of this Thread is: Adder
4
What would you like to do?
1. Addition of 2 Numbers
2. Subtraction of 2 Numbers
3. Multiplication of 2 Numbers
4. Division of 2 Numbers


4
Enter the 2 Numbers
5
0
You have chosen Division!
Starting the Thread
```

```
The Name of this Thread is: Divider
You cant Divide by Zero!
What would you like to do?
1. Addition of 2 Numbers
2. Subtraction of 2 Numbers
3. Multiplication of 2 Numbers
4. Division of 2 Numbers


5
```

Listing 2: Output for Problem 1 - calculations

7.2 Java Implementation of Problem 2

```java
// Krishnaraj Thadesar
// Batch A1, PA20
// OOPCJ Assignment 7.2
// Print even and odd numbers in increasing order using two threads in Java

import java.security.ProtectionDomain;
import java.util.Scanner;

import javax.swing.InputMap;

class printEven extends Thread implements Runnable {
    int limit;

    printEven(int limit) {
        this.limit = limit;
    }

    @Override
    public synchronized void start() {
        super.start();
        System.out.println("Printing Even Numbers");
    }

    @Override
    public void run() {
        for (int i = 0; i < limit; i++) {
            if (i % 2 == 0) {
                System.out.println(i);
            }
        }
    }
}

class printOdd extends Thread implements Runnable {
    int limit;

    printOdd(int limit) {
        this.limit = limit;
    }

    @Override
    public synchronized void start() {
        super.start();
        System.out.println("Printing Odd Numbers");
```

```
    }

    @Override
    public void run() {
        for (int i = 0; i < limit; i++) {
            if (i % 2 != 0) {
                System.out.println(i);
            }
        }
    }
}

public class assignment_7_problem_2 {
    static printEven pe;
    static printOdd po;
    static Scanner input = new Scanner(System.in);

    public static void main(String[] args) {
        int limit = 0;
        System.out.println("Enter To what limit Even or Odd numbers you want to
    See");
        limit = input.nextInt();
        pe = new printEven(limit);
        po = new printOdd(limit);
        try {
            pe.start();
            pe.join();
            po.start();
            po.join();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Listing 3: Multithreading Even Odd

7.2.1 Java Output

```
Enter To what limit Even or Odd numbers you want to See
10
Printing Even Numbers
0
2
4
6
8
Printing Odd Numbers
1
3
5
7
9
```

Listing 4: Output for ProblemHillStation 2

8 Conclusion

Thus, learnt the use of thread class in java and performed multithreading operations.

9 FAQs

1. *What are the challenges for multithreading implementation using Java Programming?*

   (a) **Difficulty of writing code**: Multithreaded and multicontexted applications are not easy to write. Only experienced programmers should undertake coding for these types of applications.

   (b) **Difficulty of debugging**: It is much harder to replicate an error in a multithreaded or multicontexted application than it is to do so in a single-threaded, single-contexted application. As a result, it is more difficult, in the former case, to identify and verify root causes when errors occur.

   (c) **Difficulty of managing concurrency**: The task of managing concurrency among threads is difficult and has the potential to introduce new problems into an application.

   (d) **Difficulty of testing**: Testing a multithreaded application is more difficult than testing a single-threaded application because defects are often timing-related and more difficult to reproduce.

   (e) **Difficulty of porting existing code**: Existing code often requires significant re-architecting to take advantage of multithreading and multicontexting. Programmers need to:

   - Remove static variables
   - Replace any function calls that are not thread-safe
   - Replace any other code that is not thread-safe

2. *What is the difference between the start and run method in Java Thread?*
   start()

   - Creates a new thread and the run() method is executed on the newly created thread.
   - Can't be invoked more than one time otherwise throws java.lang.IllegalStateException
   - Defined in java.lang.Thread class.

   run()

   - No new thread is created and the run() method is executed on the calling thread itself.
   - Multiple invocation is possible.
   - It is just a normal function in Runnable Interface.
   - Defined in java.lang.Runnable interface and must be overriden in the implementing class.

3. *Which one is better to implement thread in Java? extending Thread class or implementing Runnable?*

   (a) The significant differences between extending Thread class and implementing Runnable interface

   (b) When we extend Thread class, we cant extend any other class even we require and When we implement Runnable, we can save a space for our class to extend any other class in future or now.

(c) When we extend Thread class, each of our thread creates unique object and associate with it. When we implements Runnable, it shares the same object to multiple threads.

4. *What is the difference between wait and sleep in Java? Explain with example.*
wait():

- Wait() method belongs to Object class.
- Wait() method releases lock during Synchronization.
- Wait() should be called only from Synchronized context.
- Wait() is not a static method.
- Wait() Has Three Overloaded Methods:
  (a) wait()
  (b) wait(long timeout)
  (c) wait(long timeout, int nanos)

sleep()

- Sleep() method belongs to Thread class.
- Sleep() method does not release the lock on object during Synchronization.
- There is no need to call sleep() from Synchronized context.
- Sleep() is a static method.
- Sleep() Has Two Overloaded Methods:
  (a) sleep(long millis)millis: milliseconds
  (b) sleep(long millis,int nanos) nanos: Nanoseconds

5. *What is the difference between the submit() and execute() method of Executor and ExecutorService in Java? Explain with example.*
submit():

(a) This function executes the given command at some time in the future. The command may execute in a new thread, in a pooled thread, or in the calling thread, at the discretion of the Executor implementation.

(b) Unlike the execute method, this method returns a future. The future object is used to handle the task after the execution has started.

(c) Therefore, when we need the result of the execution, then we can use the submit() method of the future object. In order to get the result, we can use the get() methods on the Future. The get() method returns an object.

```
1         import java.util.concurrent.*;
2 public class Test {
3     public static void main(String[] args) throws Exception
4     {
5         ExecutorService executorService = Executors.newFixedThreadPool(1)
      ;
6         Future obj = executorService.submit(new Callable() {
7                   // Overriding the call method
8                   public Object call()
9                   {
10                      System.out.println(
```

```
                            "This is submit() "
                            + "method example");

                        return "Returning Callable "
                            + "Task Result";
                    }
                });
        System.out.println(obj.get());
        executorService.shutdown();
    }
}

```

execute():

(a) This function executes the given command at some time in the future. The command may execute in a new thread, in a pooled thread, or in the calling thread, at the discretion of the Executor implementation.

(b) This method is a void method meaning it doesn't return any function. Once the task is assigned in the execute() method, we won't get any response and we can forget about the task.-

```
import java.util.concurrent.*;
public class Test {

  public static void main(String[] args) throws Exception
  {

    ExecutorService executorService = Executors.newSingleThreadExecutor()
    ;
    executorService.execute(new Runnable() {
      // Override the run method
      public void run()
      {
        System.out.println(
          "This is execute() "
          + "method example");
      }
    });
    executorService.shutdown();
  }
}


```

MIT World Peace University

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 1

Developing a Simple Graphical Calculator using Swing in Java

Practical Report
Assignment 8

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

November 22, 2022

Contents

1 Aim and Objectives

Aim

To Develop a simple calculator using Swing in Java

Objective

1. To understand concept of AWT and Java swings
2. To explore Java Swing containers

2 Problem Statement

Write a Java program to create a simple calculator with the help of java swing.

3 Theory

3.1 Java Swing containers

Containers are an integral part of SWING GUI components. A container provides a space where a component can be located. A Container in AWT is a component itself and it provides the capability to add a component to itself. Following are certain noticable points to be considered.

1. Panel: JPanel is the simplest container. It provides space in which any other component can be placed, including other panels.
2. Frame: A JFrame is a top-level window with a title and a border.
3. Window: A JWindow object is a top-level window with no borders and no menubar.

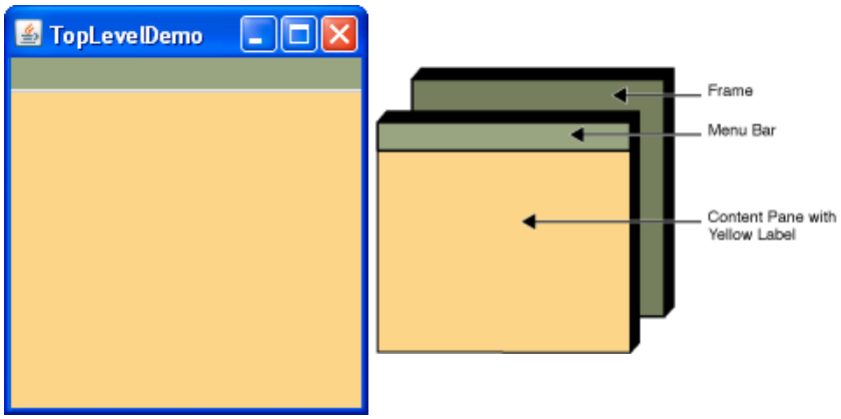


Figure 1: Top level containers in Java

3.2 Container classes of Java Swing with examples

3.2.1 JPanel

A panel is a component that is contained inside a frame window. A frame can have more than one-panel components inside it with each panel component having several other components.

```
import javax.swing.*;
class JPanelExample {
    JPanelExample(){
        JFrame frame = new JFrame("Panel Example"); //create a frame
        JPanel panel = new JPanel(); //Create JPanel Object
        panel.setBounds(40,70,100,100); //set dimensions for Panel
        JButton b = new JButton("ButtonInPanel"); //create JButton object
        b.setBounds(60,50,80,40); //set dimensions for button
        panel.add(b);   //add button to the panel
        frame.add(panel);   //add panel to frame
        frame.setSize(400,400);
        frame.setLayout(null);
        frame.setVisible(true);
    }

}
public class Main {
    public static void main(String[] args) {
      new JPanelExample(); //create an object of FrameInherited class
    }
}
```

3.2.2 JFrame

A Frame, in general, is a container that can contain other components such as buttons, labels, text fields, etc. A Frame window can contain a title, a border, and also menus, text fields, buttons, and other components. An application should contain a frame so that we can add components inside it.

The Frame in Java Swing is defined in class javax.swing.JFrame. JFrame class inherits the java.awt.Frame class. JFrame is like the main window of the GUI application using swing.

```
import javax.swing.*;
class FrameInherited extends JFrame{    //inherit from JFrame class
    JFrame f;
    FrameInherited(){
        JButton b=new JButton("JFrame_Button");//create button object
        b.setBounds(100,50,150, 40);

        add(b);//add button on frame
        setSize(300,200);
        setLayout(null);
        setVisible(true);
    }
}
public class Main {
    public static void main(String[] args) {
      new FrameInherited(); //create an object of FrameInherited class
    }
}
```

3.2.3 JWindow

The class JWindow is a container that can be displayed but does not have the title bar or window-management buttons.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class SwingContainerDemo {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;
    private JLabel msglabel;

    public SwingContainerDemo() {
        prepareGUI();
    }

    public static void main(String[] args) {
        SwingContainerDemo swingContainerDemo = new SwingContainerDemo();
        swingContainerDemo.showJWindowDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame("Java Swing Examples");
        mainFrame.setSize(400, 400);
        mainFrame.setLayout(new GridLayout(3, 1));

        mainFrame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                System.exit(0);
            }
        });
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);

        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());

        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
    }
    private void showJWindowDemo() {
        headerLabel.setText("Container in action: JWindow");
        mainFrame.setVisible(true);
    }

}
```

3.3 Swing components including buttons, checkboxes, sliders, and list boxes, etc.

There are many imporant Swing components that allow us to design GUIs. Here are some of them.

- **JTextArea**: TextArea defines an editable text field. It can have multiple lines. The swing class that defines the text area is JTextArea and it inherits the JTextComponent class.
- **JButton**: A button is a component that is used to create a push button with a name or label on it. In swing, the class that creates a labeled button is JButton. JButton inherits the Abstract-Button class. We can associate the ActionListener event to the button to make it take some action when it is pushed.
- **JList**: A list consists of multiple text items. Users can either select a single item **or** multiple items at a time. The class that implements the list in swing API is JList. JList is a descendent of the JComponent class.
- **JComboBox**: The JCombobox class shows a list of choices from which a user can select an option. The selected choice is at the top. JComboBox derives from the JComponent class.
- **JSlider**: A slider allows us to select a specific range of values. In Java Swing API, JSlider is the class that is used to implement the slider.
- **JCheckBox**: The JcheckBox class is used to create chekbox in swing framework.
- **JRadioButton**: Radio button is a group of related button in which only one can be selected. JRadioButton class is used to create a radio button in Frames.
- **JLabel**: In Java, Swingtoolkit contains a JLabel Class. It is under package javax.swing.JLabel class. It is used for placing text in a box. Only Single line text is allowed and the text can not be changed directly.
- **JPasswordField**: In Java, Swing toolkit contains a JPasswordField Class. It is under package javax.swing.JPasswordField class. It is specifically used for password and it can be edited.

3.4 Heavyweight Components and Lightweight Components

There are two kinds of graphics components in the Java programming language: heavyweight and lightweight.

A component is said to be a heavyweight component if it uses native code provided by your computer's operating system to display buttons, choice lists, text fields, and the like. Such operating-system routines are said to be the components's peers.

A Swing component is said to be a lightweight component because it written entirely in Java and does the high-level display work itself, rather than relying on code provided by your computer's operating system.

3.5 What is Double Buffering?

1. Double-buffering is the process of drawing graphics into an off-screen image buffer and then copying the contents of the buffer to the screen all at once.
2. For the complex graphics, using double-buffering can reduce flickering issues.
3. Java Swing automatically supports double-buffering for all of its components.

4. Double-buffering is memory intensive, its use is only justified for components that are repainted very frequently or have particularly complex graphics to display.

5. If a container uses double-buffering, any double-buffered children it has shared the off-screen buffer of the container, the required off-screen buffer is never larger than the on-screen size of the application.

6. To enable double buffering, simply call the setDoubleBuffered() method (inherited from JComponent) to set the double-buffered property to true for any components that should use double-buffered drawing.

3.6 Difference between applet and Swing

Applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. An applet is embedded in an HTML page using the APPLET or OBJECT tag and hosted on a web server. They are used to make the web site more dynamic and entertaining. All applets are sub-classes (either directly or indirectly) of java.applet.Applet class. Applets are not stand-alone programs. Instead, they run within either a web browser or an applet viewer. JDK provides a standard applet viewer tool called applet viewer.

Swing in Java is a lightweight GUI toolkit which has a wide variety of widgets for building optimized window based applications. It is a part of the JFC(Java Foundation Classes). It is build on top of the AWT API and entirely written in java.

It is platform independent unlike AWT and has lightweight components. It becomes easier to build applications since we already have GUI components like button, checkbox etc. This is helpful because we do not have to start from the scratch.

4 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers : g++ and gcc on linux for C++, and javac, with JDK 18.0.2 for Java

5 Input

The numbers and the Operators.

6 Output

The Output of the entered Calculation in the Display Section of the Calculator

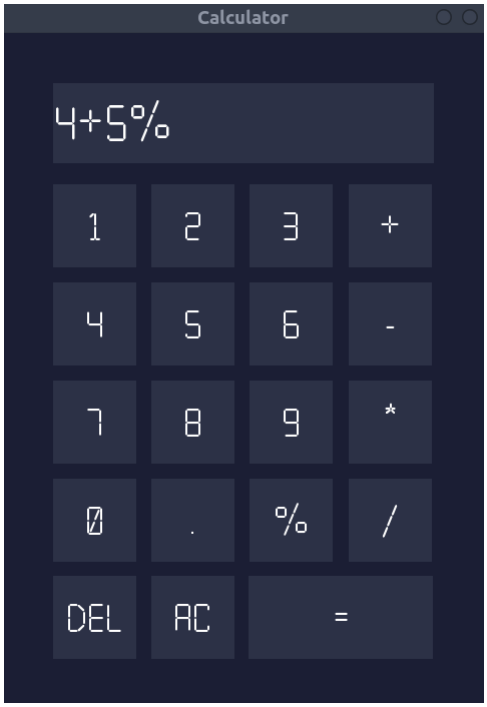


Figure 2: Calculator with Java Swing

7 Code

```
// Krishnaraj Thadesar
// Batch A1, PA20
// OOPCJ Assignment 9
// Making a Calculator in Java using Swing

package org.OOPCJ.Krishnaraj;
import org.mariuszgromada.math.mxparser.*;

import javax.swing.*;
import java.awt.*;
import java.io.*;

class Colors {
    static Color primaryColor = new Color(255, 255, 255); // text color
    static Color bgColor = new Color(27, 30, 52); // background
    static Color secondaryColor = new Color(44, 49, 70); // upper background
    static Color secondaryColorRollover = new Color(53, 59, 80); // upper
    background
    static Color accentColor = new Color(26, 122, 230); // Accent
}

class Numpad extends JPanel {
    JButton[] numbers = new JButton[12];

    Numpad() {
        this.setFocusable(true);
        this.setVisible(true);
        this.setBorder(null);
```

```
28          this.setBounds(50, 150, 280, 380);
29          this.setBackground(Colors.bgColor);
30          this.setLayout(new GridLayout(4, 3, 15, 15));
31          createButtons();
32          for (int i = 0; i < numbers.length; i++) {
33              this.add(numbers[i]);
34          }
35      }
36
37      public void createButtons() {
38          for (int i = 0; i < 12; i++) {
39              numbers[i] = new JButton();
40              numbers[i].setText(String.valueOf(i + 1));
41              numbers[i].setFocusPainted(false);
42              numbers[i].setContentAreaFilled(false);
43              numbers[i].setOpaque(true);
44              numbers[i].setBorder(null);
45              numbers[i].setBackground(Colors.secondaryColor);
46              numbers[i].setForeground(Colors.primaryColor);
47              numbers[i].setFont(Calculator.buttonFont);
48              final JButton temp = numbers[i];
49              temp.addChangeListener(evt -> {
50                  if (temp.getModel().isPressed()) {
51                      temp.setForeground(Colors.primaryColor);
52                      temp.setBackground(Colors.secondaryColorRollover);
53                  } else if (temp.getModel().isRollover()) {
54                      temp.setForeground(Colors.accentColor);
55                      temp.setBackground(Colors.secondaryColorRollover);
56                  } else {
57                      temp.setForeground(Colors.primaryColor);
58                      temp.setBackground(Colors.secondaryColor);
59                  }
60              });
61              temp.addActionListener(e -> {
62                  Calculator.display.setText(Calculator.display.getText() + ((
    JButton) e.getSource()).getText());
63              });
64          }
65          numbers[9].setText("0");
66          numbers[10].setText(".");
67          numbers[11].setText("%");
68      }
69  }
70
71  class Operators_pnl extends JPanel {
72      static JButton[] operators = new JButton[4];
73      static String currentOperator;
74
75      Operators_pnl() {
76          this.setFocusable(true);
77          this.setVisible(true);
78          this.setBorder(null);
79          this.setBounds(345, 150, (int) 250 / 3, 380);
80          this.setBackground(Colors.bgColor);
81          this.setLayout(new GridLayout(4, 1, 0, 15));
82          createButtons();
83          for (int i = 0; i < operators.length; i++) {
84              this.add(operators[i]);
85          }
```

```
86      }
87
88      public void createButtons() {
89          for (int i = 0; i < 4; i++) {
90              operators[i] = new JButton();
91              operators[i].setFocusPainted(false);
92              operators[i].setContentAreaFilled(false);
93              operators[i].setOpaque(true);
94              operators[i].setBorder(null);
95              operators[i].setBackground(Colors.secondaryColor);
96              operators[i].setForeground(Colors.primaryColor);
97              operators[i].setFont(Calculator.buttonFont);
98              final JButton temp = operators[i];
99              temp.addChangeListener(evt -> {
100                 if (temp.getModel().isPressed()) {
101                     temp.setForeground(Colors.primaryColor);
102                     temp.setBackground(Colors.secondaryColorRollover);
103                 } else if (temp.getModel().isRollover()) {
104                     temp.setForeground(Colors.accentColor);
105                     temp.setBackground(Colors.secondaryColorRollover);
106                 } else {
107                     temp.setForeground(Colors.primaryColor);
108                     temp.setBackground(Colors.secondaryColor);
109                 }
110             });
111             temp.addActionListener(e -> {
112                 if (!Calculator.operator_used) {
113                     Calculator.display.setText(Calculator.display.getText() + ((
    JButton) e.getSource()).getText());
114                 }
115             });
116         }
117         operators[0].setText("+");
118         operators[1].setText("-");
119         operators[2].setText("*");
120         operators[3].setText("/");
121     }
122 }
123
124 // Main Calculator Frame no panels
125 class Calculator extends JFrame {
126     static double number_1, number_2;
127     static boolean operator_used = false;
128     JButton clearBtn, backspaceBtn, resultBtn;
129     static JTextField display;
130     Numpad numpad;
131     Operators_pnl operators_pnl;
132     static Font buttonFont;
133
134     Calculator() {
135         this.setTitle("Calculator");
136         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
137         this.setResizable(false);
138         this.setUndecorated(false);
139         this.setPreferredSize(new Dimension(480, 670));
140         this.getContentPane().setBackground(Colors.bgColor);
141         this.setLayout(null);
142         createFonts();
143         createPanels();
```

```
144         createButtons();
145         this.add(display);
146         this.add(numpad);
147         this.add(operators_pnl);
148         this.add(clearBtn);
149         this.add(backspaceBtn);
150         this.add(resultBtn);
151         this.pack();
152         this.setVisible(true);
153         this.setLocationRelativeTo(null);
154     }
155 
156     public void createPanels() {
157         numpad = new Numpad();
158         operators_pnl = new Operators_pnl();
159     }
160 
161     public void createButtons() {
162         display = new JTextField();
163         display.setBounds(50, 50, 380, 80);
164         display.setOpaque(true);
165         display.setAlignmentX(RIGHT_ALIGNMENT);
166         display.setBorder(null);
167         display.setBackground(Colors.secondaryColor);
168         display.setForeground(Colors.primaryColor);
169         display.setFont(Calculator.buttonFont.deriveFont(55f));
170         display.addActionListener(e -> {
171         });
172 
173         clearBtn = new JButton();
174         clearBtn.setText("AC");
175         clearBtn.setBounds(50 + 15 + (int) 250 / 3, 542, (int) 250 / 3, (int) 250
    / 3);
176         clearBtn.setFocusPainted(false);
177         clearBtn.setContentAreaFilled(false);
178         clearBtn.setOpaque(true);
179         clearBtn.setBorder(null);
180         clearBtn.setBackground(Colors.secondaryColor);
181         clearBtn.setForeground(Colors.primaryColor);
182         clearBtn.setFont(Calculator.buttonFont);
183         clearBtn.addChangeListener(evt -> {
184             if (clearBtn.getModel().isPressed()) {
185                 clearBtn.setForeground(Colors.primaryColor);
186                 clearBtn.setBackground(Colors.secondaryColorRollover);
187             } else if (clearBtn.getModel().isRollover()) {
188                 clearBtn.setForeground(Colors.accentColor);
189                 clearBtn.setBackground(Colors.secondaryColorRollover);
190             } else {
191                 clearBtn.setForeground(Colors.primaryColor);
192                 clearBtn.setBackground(Colors.secondaryColor);
193             }
194         });
195         clearBtn.addActionListener(e -> {
196             display.setText("");
197             operator_used = false;
198             Operators_pnl.operators[0].setEnabled(true);
199             Operators_pnl.operators[1].setEnabled(true);
200             Operators_pnl.operators[2].setEnabled(true);
201             Operators_pnl.operators[3].setEnabled(true);
```

```
202         });
203
204         backspaceBtn = new JButton();
205         backspaceBtn.setText("DEL");
206         backspaceBtn.setBounds(50, 542, (int) 250 / 3, (int) 250 / 3);
207         backspaceBtn.setFocusPainted(false);
208         backspaceBtn.setContentAreaFilled(false);
209         backspaceBtn.setOpaque(true);
210         backspaceBtn.setBorder(null);
211         backspaceBtn.setBackground(Colors.secondaryColor);
212         backspaceBtn.setForeground(Colors.primaryColor);
213         backspaceBtn.setFont(Calculator.buttonFont);
214         backspaceBtn.addChangeListener(evt -> {
215             if (backspaceBtn.getModel().isPressed()) {
216                 backspaceBtn.setForeground(Colors.primaryColor);
217                 backspaceBtn.setBackground(Colors.secondaryColorRollover);
218             } else if (backspaceBtn.getModel().isRollover()) {
219                 backspaceBtn.setForeground(Colors.accentColor);
220                 backspaceBtn.setBackground(Colors.secondaryColorRollover);
221             } else {
222                 backspaceBtn.setForeground(Colors.primaryColor);
223                 backspaceBtn.setBackground(Colors.secondaryColor);
224             }
225         });
226         backspaceBtn.addActionListener(e -> {
227             try {
228                 display.setText(display.getText().substring(0, display.getText().
    length() - 1));
229             } catch (Exception f) {
230                 System.out.println("You got nothing on screen then how can you
    delete? ");
231             }
232         });
233
234         resultBtn = new JButton();
235         resultBtn.setText("=");
236         resultBtn.setBounds(245, 542, (int) 184, (int) 250 / 3);
237         resultBtn.setFocusPainted(false);
238         resultBtn.setContentAreaFilled(false);
239         resultBtn.setOpaque(true);
240         resultBtn.setBorder(null);
241         resultBtn.setBackground(Colors.secondaryColor);
242         resultBtn.setForeground(Colors.primaryColor);
243         resultBtn.setFont(Calculator.buttonFont);
244         resultBtn.addChangeListener(evt -> {
245             if (resultBtn.getModel().isPressed()) {
246                 resultBtn.setForeground(Colors.primaryColor);
247                 resultBtn.setBackground(Colors.secondaryColorRollover);
248             } else if (resultBtn.getModel().isRollover()) {
249                 resultBtn.setForeground(Colors.accentColor);
250                 resultBtn.setBackground(Colors.secondaryColorRollover);
251             } else {
252                 resultBtn.setForeground(Colors.primaryColor);
253                 resultBtn.setBackground(Colors.secondaryColor);
254             }
255         });
256         resultBtn.addActionListener(e -> {
257             String currentString = display.getText();
258             Expression expr = new Expression(currentString);
```

```
            display.setText(String.valueOf(expr.calculate()));
        });
    }

    public static void createFonts() {
        try {
            buttonFont = Font.createFont(Font.TRUETYPE_FONT, new File("/run/media/krishnaraj/Classes/University/Second Year/First Semister/OOPJC/Programs/java_implementations/assignment_8/Calculator/src/main/resources/Calculator.ttf")).deriveFont(45f);
            GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
            // register the font
            ge.registerFont(buttonFont);

        } catch (FontFormatException | IOException e) {
            e.printStackTrace();
        }
    }
}

public class Main {
    static Calculator calc;

    public static void main(String[] args) {
        calc = new Calculator();
    }
}
```

Listing 1: Calculator.java

8 Dependencies

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>org.OOPCJ.Krishnaraj.Calculator</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>18</maven.compiler.source>
        <maven.compiler.target>18</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.mariuszgromada.math</groupId>
            <artifactId>MathParser.org-mXparser</artifactId>
            <version>5.0.7</version>
        </dependency>

    </dependencies>

```

```
</project>
```

Listing 2: pom.xml

9 Conclusion

Thus, implemented simple calculator with the help of java swing and performed various operations.

10 FAQs

1. *Methods of component class in Java Swing?*

   A component is an object having a graphical representation that can be displayed on the screen and that can interact with the user. Examples of components are the buttons, checkboxes, and scrollbars of a typical graphical user interface.

   The Component class is the abstract superclass of the nonmenu-related Abstract Window Toolkit components. Class Component can also be extended directly to create a lightweight component. A lightweight component is a component that is not associated with a native window. On the contrary, a heavyweight component is associated with a native window. The isLightweight() method may be used to distinguish between the two kinds of the components.

   Some methods of this class are:

   - **void add(PopupMenu):** Adds the specified popup menu to the component.
   - **void addComponentListener(ComponentListener):** Adds the specified component listener to receive component events from this component.
   - **void addFocusListener(FocusListener):** Adds the specified focus listener to receive focus events from this component when this component gains input focus.
   - **Rectangle getBounds(): Gets** the bounds of this component in the form of a Rectangle object.
   - **Rectangle getBounds(Rectangle):** Stores the bounds of this component into "return value" rv and return rv.

2. *Ways to create a frame in Java Swing? Explain with examples*

   (a) By creating the object of Frame class (association)

```
// Java program to create frames
// using association

import javax.swing.*;
public class test1
{
  JFrame frame;

  test1()
  {
    // creating instance of JFrame with name "first way"
    frame=new JFrame("first way");

    // creates instance of JButton
```

```
    JButton button = new JButton("let's see");

    button.setBounds(200, 150, 90, 50);

    // setting close operation
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // adds button in JFrame
    frame.add(button);

    // sets 500 width and 600 height
    frame.setSize(500, 600);

    // uses no layout managers
    frame.setLayout(null);

    // makes the frame visible
    frame.setVisible(true);
  }

  public static void main(String[] args)
  {
    new test1();
  }
}
```

(b) By extending Frame class (inheritance)

```
// Java program to create a
// frame using inheritance().

import javax.swing.*;

// inheriting JFrame
public class test2 extends JFrame
{
  JFrame frame;
  test2()
  {
    setTitle("this is also a title");

    // create button
    JButton button = new JButton("click");

    button.setBounds(165, 135, 115, 55);

    // adding button on frame
    add(button);

    // setting close operation
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    setSize(400, 500);
    setLayout(null);
    setVisible(true);
  }

  public static void main(String[] args)
  {
```

```
    new test2();
  }
}


```

(c) Create a frame using Swing inside main()

```
// Java program to create a frame
// using Swings in main().

import javax.swing.*;
public class Swing_example
{
  public static void main(String[] args)
  {
    // creates instance of JFrame
    JFrame frame1 = new JFrame();

    // creates instance of JButton
    JButton button1 = new JButton("click");
    JButton button2 = new JButton("again click");

    // x axis, y axis, width, height
    button1.setBounds(160, 150 ,80, 80);
    button2.setBounds(190, 190, 100, 200);

    // adds button1 in Frame1
    frame1.add(button1);

    // adds button2 in Frame1
    frame1.add(button2);

    // 400 width and 500 height of frame1
    frame1.setSize(400, 500) ;

    // uses no layout managers
    frame1.setLayout(null);

    // makes the frame visible
    frame1.setVisible(true);
  }
}


```

3. *What are the Methods of JLabel class in Java Swing?*

- **String getText()**: Returns the text string that the label displays.
- **LabelUI getUI()**: Returns the LF object that renders this component.
- **String getUIClassID()**: Returns a string that specifies the name of the lf class that renders this component.
- **void setIcon(Icon icon)**: Defines the icon this component will display.
- **void setIconTextGap(int iconTextGap)**: If both the icon and text properties are set, this property defines the space between them.
- **void setLabelFor(Component c)**: Set the component this is labelling.

- **void setText(String text)**: Defines the single line of text this component will display.
- **void setUI(LabelUI ui)**: Sets the LF object that renders this component.
- **void setVerticalAlignment(int alignment)**: Sets the alignment of the label's contents along the Y axis.
- **void setVerticalTextPosition(int textPosition)**: Sets the vertical position of the label's text, relative to its image.
- **void updateUI()**: Resets the UI property to a value from the current look and feel.

4. *What are the Methods of AbstractButton class in Java Swing?*

- **protected ActionListener actionListener**: The button model's ActionListener.
- **protected ChangeEvent changeEvent**: Only one ChangeEvent is needed per button instance since the event's only state is the source property.
- **protected ChangeListener changeListener**: The button model's changeListener.

5. *Write a simple Java Swing program of displaying image on the button?*

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

class test extends JFrame {

    test() {
        JButton bt1 = new JButton("no"); // Creating a Yes Button.
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // setting close
    operation.
        bt1.setBounds(60, 50, 500, 500);
        ImageIcon imageIcon = new ImageIcon("../../Lab/pic.jpg");
        bt1.setIcon(imageIcon);
        setLayout(null); // setting layout using FlowLayout object
        setSize(700, 700); // setting size of Jframe
        add(bt1); // adding Yes button to frame.
        setVisible(true);
    }

    public static void main(String[] args) {
        new test();
    }
}
```

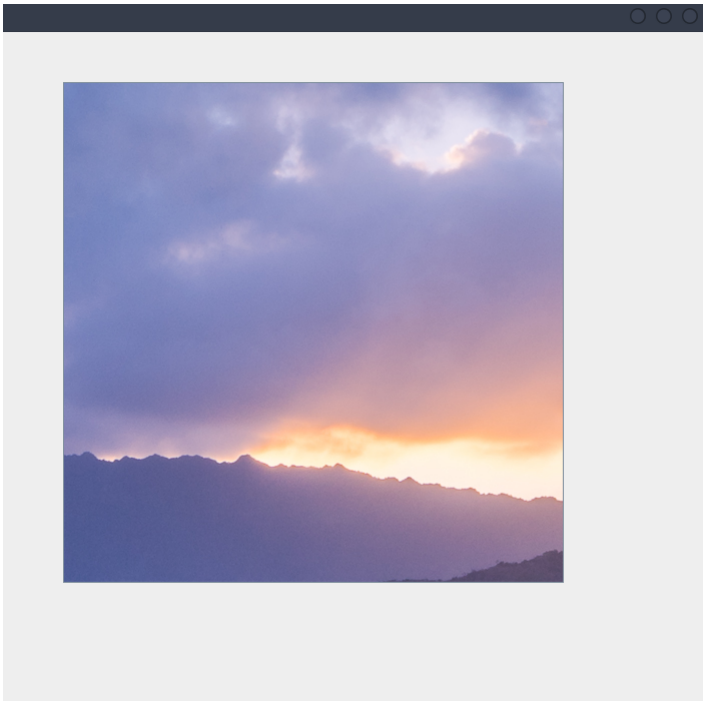


Figure 3: Button with a background image in its background.

MIT World Peace University

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 1

Applet Using Java and HTML

Practical Report
Assignment 9

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

November 28, 2022

Contents

1 Aim and Objectives

Aim

Develop an applet that displays a simple message in centre of the screen

Objectives

1. To understand concept of Java // Krishnaraj Thadesar // Batch A1, PA20 // OOPCJ Assignment 9Applets
2. To explore features of applets to develop web applications

2 Problem Statements

Write a Java applet program that displays a simple message in centre of the screen.

3 Theory

3.1 Introduction to an Applet and its features

Java applets were small applications written in the Java programming language, or another programming language that compiles to Java bytecode, and delivered to users in the form of Java bytecode. The user launched the Java applet from a web page, and the applet was then executed within a Java virtual machine (JVM) in a process separate from the web browser itself. A Java applet could appear in a frame of the web page, a new application window, Sun's AppletViewer, or a stand-alone tool for testing applets.

Java applets were introduced in the first version of the Java language, which was released in 1995. Beginning in 2013, major web browsers began to phase out support for the underlying technology applets used to run, with applets becoming completely unable to be run by 2015-2017. Java applets were deprecated by Java 9 in 2017

An applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. An applet is embedded in an HTML page using the APPLET or OBJECT tag and hosted on a web server. Applets are used to make the website more dynamic and entertaining.

Here are some important points:

- All applets are sub-classes (either directly or indirectly) of java.applet.Applet class.
- Applets are not stand-alone programs. Instead, they run within either a web browser or an applet viewer. JDK provides a standard applet viewer tool called applet viewer.
- In general, execution of an applet does not begin at main() method.
- Output of an applet window is not performed by System.out.println(). Rather it is handled with various AWT methods, such as drawString().

3.2 The Lifecycle of an Applet

Five methods in the Applet class gives you the framework on which you build any serious applet.

1. *init* - This method is intended for whatever initialization is needed for your applet. It is called after the param tags inside the applet tag have been processed.
2. *start* - This method is automatically called after the browser calls the init method. It is also called whenever the user returns to the page containing the applet after having gone off to other pages.
3. *stop* - This method is automatically called when the user moves off the page on which the applet sits. It can, therefore, be called repeatedly in the same applet.
4. *destroy* - This method is only called when the browser shuts down normally. Because applets are meant to live on an HTML page, you should not normally leave resources behind after a user leaves the page that contains the applet.
5. *paint* - Invoked immediately after the start() method, and also any time the applet needs to repaint itself in the browser. The paint() method is actually inherited from the java.awt.

3.3 Discuss about applet tag and its importance

An applet may be invoked by embedding directives in an HTML file and viewing the file through an applet viewer or Java-enabled browser. The <applet> tag is the basis for embedding an applet in an HTML file.

```
<!DOCTYPE html>
<html>
   <head>
    <title>HTML applet Tag</title>
   </head>
   <body>
    <applet code = "newClass.class" width = "300" height = "200"></applet>
   </body>
</html>
```

It is now only supported on Internet Exporer, Firefox, and Safari. Here is an example using the above attributes in the applet tag.

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML applet Tag</title>
  </head>
  <body>
    <applet
      code="Demo.class"
      width="300"
      height="200"
      alt="It is a class"
      align="left"
      name="New Applet"
      title="A Title for the Applet"
      vspace="30px"
    ></applet>
  </body>
</html>
```

3.4 Explain various methods of Applet class with necessary examples.

Attribute	Value	Description
align	URL	*Deprecated - Defines the text alignment around the applet*
alt	URL	Alternate text to be displayed in case browser does not support applet
archive	URL	Applet path when it is stored in a Java Archive ie. jar file
code	URL	A URL that points to the class of the applet
codebase	URL	Indicates the base URL of the applet if the code attribute is relative
height	pixels	Height to display the applet
hspace	pixels	*Deprecated - Defines the left and right spacing around the applet*
name	name	Defines a unique name for the applet
object	name	Specifies the resource that contains a serialized representation of the applet's state.
title	test	Additional information to be displayed in tool tip of the mouse
vspace	pixels	*Deprecated - Amount of white space to be inserted above and below the object.*
width	pixels	Width to display the applet.

4 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers : g++ and gcc on linux for C++, and javac, with JDK 18.0.2 for Java

5 Output

The Applet with some text written on it being displayed on the screen.

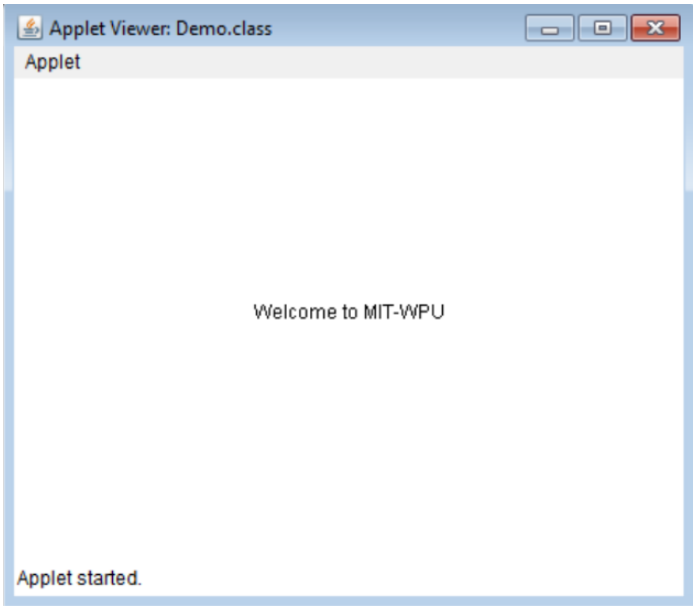


Figure 1:

6 Code

```java
// Krishnaraj Thadesar
// Batch A1, PA20
// OOPCJ Assignment 9
// Write a Java applet program that displays a simple message in centre of the
    screen.


import java.applet.*;
import java.awt.*;

public class Assignment_9 extends Applet {
    public void print(Graphics g) {
        g.drawString("Welcome to Java Applets in Assignment 9", 150, 150);
    }
}
```

Listing 1: applet.java

```html
<!-- <!DOCTYPE html> -->
<html>
  <head>
    <title>Assignment 9</title>
  </head>
  <body>
    <applet code="Assignment_9.class" width="300" height="300"></applet>
  </body>
</html>
```

Listing 2: applet.html

7 Conclusion

Thus, developed an applet that displays a simple message in centre of the screen.

8 FAQs

1. *What are the restrictions imposed on Java applets?* Mostly due to security reasons, the following restrictions are imposed on Java applets:

   (a) An applet cannot load libraries or define native methods.

   (b) An applet cannot ordinarily read or write files on the execution host.

   (c) An applet cannot read certain system properties.

   (d) An applet cannot make network connections except to the host that it came from.

   (e) An applet cannot start any program on the host that's executing it.

   (f)

2. *What is the applet class loader, and what does it provide?*

   The class loader is the means by which Java classes and resources are loaded into the JRE. It controls the policies ranging from where to load class definitions to the data format of the class definitions.

A system class loader responsible for loading in the Java runtime, the application, and classes and resources in the application's classpath. An applet class loader is responsible for loading the applets and their related classes and resources, possibly over the network by communicating with a Web serve

When an applet is loaded over the internet, the applet is loaded by the applet classloader. The class loader enforces the Java name space hierarchy. Also, the class loader guarantees that a unique namespace exists for classes that come from the local file system, and that a unique namespace exists for each network source.

When a browser loads an applet over the net, that applet's classes are placed in a private namespace associated with the applet's origin. Then, those classes loaded by the class loader are passed through the verifier.

3. *What is the applet security manager, and what does it provide?* A security manager is an object that defines a security policy for an application. This policy specifies actions that are unsafe or sensitive. Any actions not allowed by the security policy cause a SecurityException to be thrown. An application can also query its security manager to discover which actions are allowed.

   Typically, a web applet runs with a security manager provided by the browser or Java Web Start plugin. Other kinds of applications normally run without a security manager, unless the application itself defines one. If no security manager is present, the application has no security policy and acts without restrictions.

4. *Explain the following with suitable examples*

   (a) **Creating an applet**

```java
import java.applet.*;
import java.awt.*;

public class Assignment_9 extends Applet {
    public void print(Graphics g) {
        g.drawString("Welcome to Java Applets in Assignment 9", 150, 150)
    ;
    }
}

```

   (b) **Passing parameters to applets**

```java
import java.applet.Applet;
import java.awt.Graphics;

public class UseParam extends Applet{

  public void paint(Graphics g){
  String str=getParameter("msg");
  g.drawString(str,50, 50);
  }

}
```

```
<html>
<body>
<applet code="UseParam.class" width="300" height="300">
<param name="msg" value="Welcome to applet">
</applet>
</body>
</html>
```

(c) Adding graphics and colors to applets.

```
import java.applet.Applet;
import java.awt.*;

public class GraphicsDemo extends Applet{

  public void paint(Graphics g){
    g.setColor(Color.red);
    g.drawString("Welcome",50, 50);
    g.drawLine(20,30,20,300);
    g.drawRect(70,100,30,30);
    g.fillRect(170,100,30,30);
    g.drawOval(70,200,30,30);

    g.setColor(Color.pink);
    g.fillOval(170,200,30,30);
    g.drawArc(90,150,30,30,30,270);
    g.fillArc(270,150,30,30,0,180);
  }
}

```

```
<html>
<body>
<applet code="GraphicsDemo.class" width="300" height="300">
</applet>
</body>
</html>

```

MIT World Peace University

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 1

Theory Assignment

Assignment No. 1

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 20

October 9, 2022

1 Questions

1.1 Explain the Various Features of Object Oriented Programming, and also note down its applications in various domains.

1. **Encapsulation Enforces Modularity**
   Encapsulation refers to the creation of self-contained modules that bind processing functions to the data. These user-defined data types are called "classes," and one instance of a class is an "object." For example, in a payroll system, a class could be Manager, and Pat and Jan could be two instances (two objects) of the Manager class. Encapsulation ensures good code modularity, which keeps routines separate and less prone to conflict with each other.

2. **Inheritance Passes "Knowledge" Down**
   Classes are created in hierarchies, and inheritance allows the structure and methods in one class to be passed down the hierarchy. That means less programming is required when adding functions to complex systems. If a step is added at the bottom of a hierarchy, only the processing and data associated with that unique step needs to be added. Everything else is inherited. The ability to reuse existing objects is considered a major advantage of object technology.

3. **Polymorphism Takes any Shape**
   Object-oriented programming allows procedures about objects to be created whose exact type is not known until runtime. For example, a screen cursor may change its shape from an arrow to a line depending on the program mode. The routine to move the cursor on screen in response to mouse movement would be written for "cursor," and polymorphism allows that cursor to take on whatever shape is required at runtime. It also allows new shapes to be easily integrated.

4. **Data Abstraction Hides the Unneccessory**
   Abstraction refers to the user's interaction with just a subset of an object's characteristics and operations. To access a complicated item, abstraction uses simpler, high-level techniques.

   Simple items are used to show complexity. Keep complicated information hidden from the user. Simple classes are used to indicate complexity in abstraction. Encapsulation is an extension of abstraction.

Some Applications of OOP Languages in Various Domains are given below

1. **Real Time Systems** The term "real-time system" refers to any information processing system with hardware and software components that perform real-time application functions and can respond to events within predictable and specific time constraints. Using Object-oriented technology, we can develop real-time systems, this will offer adaptability, ease of modifications, reusability for the code. There is a lot of complexity involved in designing real-time systems, OOP techniques make it easier to handle those complexities.

2. **Client Server System**

   The client-server systems are those that involve a relationship between cooperating programs in an application. In general, the clients will initiate requests for services and the servers will provide that functionality. The client and server either reside in the same system or communicate with each other through a computer network or the internet.

3. **Object Oriented Database**

   Nowadays each and every data is being stored and processed, the traditional model of storing data i.e the relational model stores each and every piece of data in tables that consist of rows and columns. However as complexity grows, storing in the form of tables becomes quite cumbersome, here the need for storing in the form of real-world objects comes into the picture. These databases try to maintain a direct correspondence between the real-world and database objects in order to let the object retain its identity and integrity. They can then be identified and operated upon.

   A popular example of object-oriented databases is **MongoDB**.

4. **Neural Networks and Parallel Programming** A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature.

5. **Simulation and Modeling System**

   A good example of a simulation and modeling system is of automobiles such as cars, Once the model of the car structure is developed by the engineer's team, as and when they feel that the product is good to go they can release the product. OOP provides an appropriate approach for simplifying these complex models.

6. **CIM/CAD/CAM Systems** OOP can also be used in manufacturing and designing applications as it allows people to reduce the efforts involved. For instance, it can be used while designing blueprints and flowcharts. So it makes it possible to produce these flowcharts and blueprint accurately.

7. **Computer Aided Designs**

   As per Wikipedia, Computer-aided design (CAD) is the use of computers (or workstations) to aid in the creation, modification, analysis, or optimization of a design. In mechanical design, it is known as mechanical design automation (MDA), which includes the process of creating a technical drawing with the use of computer software.

   One of the good examples of computer-aided design is **MATLAB**. It is used by the programmers to solve difficult mathematical models, these models are then used in bigger system designs to check whether the system will function as expected or not

1.2 Explain difference between compile time polymorphism and run time polymorphism.

Compile Time Polymorphism

In compile-time polymorphism, a function is called at the time of program compilation. We call this type of polymorphism as early binding or Static binding. Function overloading and operator overloading is the type of Compile time polymorphism. It can be implemented in 2 simple ways in c++

1. **Function Overloading** :
   Function overloading means one function can perform many tasks. In C++, a single function is used to perform many tasks with the same name and different types of arguments. In the

function overloading function will call at the time of program compilation. It is an example of compile-time polymorphism.

```
class Addition {
public:
  int ADD(int X,int Y)   // Function with parameter
  {
    return X+Y;     // this function is performing addition of  two Integer value
  }
  int ADD() {              // Function with same name but without parameter
    string a= "HELLO";
    string b="SAM";   // in this function concatenation is performed
  string c= a+b;
  cout<<c<<endl;

  }
};
int main(void) {
  Addition obj;   // Object is created
  cout<<obj.ADD(128, 15)<<endl; //first method is called
  obj.ADD();  // second method is called
  return 0;
}


```

2. **Operator Overloading**:
   Operator overloading means defining additional tasks to operators without changing its actual meaning. We do this by using operator function.
   ◦ The purpose of operator overloading is to provide a special meaning to the user-defined data types.
   The advantage of Operators overloading is to perform different operations on the same operand.

```
  #include <iostream>
  using namespace std;
  class A
  {

    string x;
    public:
    A(){}
    A(string i)
    {
    x=i;
    }
    void operator+(A);
    void display();
  };

  void A:: operator+(A a)
  {

    string m = x+a.x;
    cout<<"The result of the addition of two objects is : "<<m;

```

```
      }
      int main()
      {
        A a1("Welcome");
        A a2("back");
        a1+a2;
        return 0;
      }


```

Run Time Polymorphism

In Runtime polymorphism, functions are called at the time the program execution. Hence, it is known as late binding or dynamic binding.

Function overriding is a part of runtime polymorphism. In function overriding, more than one method has the same name with different types of the parameter list.

It is achieved by using virtual functions and pointers. It provides slow execution as it is known at the run time. Thus, It is more flexible as all the things executed at the run time.

In C++ it can be implemented using these 2 methods.

1. **Function Overriding:**
   In function overriding, we give the new definition to base class function in the derived class. At that time, we can say the base function has been overridden. It can be only possible in the 'derived class'. In function overriding, we have two definitions of the same function, one in the superclass and one in the derived class. The decision about which function definition requires calling happens at runtime. That is the reason we call it 'Runtime polymorphism'.

```

// Java program to demonstrate
// runtime polymorphism

// Implementing a class
class Test {

    // Implementing a method
    public void method()
    {
        System.out.println("Method 1");
    }
}

// Defining a child class
public class GFG extends Test {

    // Overriding the parent method
    public void method()
    {
        System.out.println("Method 2");
    }

    // Driver code
    public static void main(String args[])
    {
        Test test = new GFG();

```

```
        test.method();
    }
}

```

2. **Virtual Functions:**
   A virtual function is declared by keyword virtual. The return type of virtual function may be int, float, void.

   A virtual function is a member function in the base class. We can redefine it in a derived class. It is part of run time polymorphism. The declaration of the virtual function must be in the base class by using the keyword virtual. A virtual function is not static.

   The virtual function helps to tell the compiler to perform dynamic binding or late binding on the function.

   If it is necessary to use a single pointer to refer to all the different classes' objects. This is because we will have to create a pointer to the base class that refers to all the derived objects.

```
#include <iostream>
using namespace std;
class Base
{
public:
  virtual void show_val()
  {
    cout << "Class::Base";
  }
  };
  class Derived : public Base
  {
    public:
  void show_val()
  {
    cout << "Class::Derived";
    }
    };
    int main()
    {
      Base *b; // Base class pointer Derived d; //Derived class object b = &d;
    b->show_val();   //late Binding
}
```

1.3 Create a Java program to find the factorial of a number using default constructor and parameterized constructor.

```

// Code to find factorial by using default and parameterized constructor.
import java.util.*;

class Factorial {
    int n = 0;
    Scanner input = new Scanner(System.in);

    Factorial() {
        System.out.println("Enter a Number whose factorial you want to find: ");
        this.n = input.nextInt();
```

```
    }

    Factorial(int n) {
        this.n = n;
    }

    int factorial() {
        int res = 1, i;
        for (i = 2; i <= this.n; i++)
            res *= i;
        return res;
    }
}

class HelloWorld {
    // Driver Code
    public static void main(String args[]) {
        Factorial para = new Factorial(5);
        Factorial def = new Factorial();

        System.out.println("(Parameterized) Factorial : " +
                "  is " + para.factorial());
        System.out.println("\n(Default) Factorial :   " +
                "  is " + def.factorial());

    }
}
```

```
Enter a Number whose factorial you want to find: 6
(Parameterized) Factorial :  is 120
(Default) Factorial :   is 720
```

1.4 Create a C++ program that creates a class "Arithmatic" which contains integer data members. Overload all the four arthmetic operators so that they operate on the objects of "Arthmatic".

```
// Create a C++ program that creates a class "Arithmatic" which contains integer
   data members. Overload all the four arthmetic operators so that they operate
  on the objects of "Arthmatic".

#include <iostream>
using namespace std;

class Arithmatic
{
public:
  int x, y;
  Arithmatic(int xx = 0, int yy = 0)
  {
    x = xx;
    y = yy;
  }

  Arithmatic operator+(Arithmatic const &obj)
  {
    Arithmatic res;
```

```
      res.x = x + obj.x;
      res.y = y + obj.y;
      return res;
    }
    Arithmatic operator-(Arithmatic const &obj)
    {
      Arithmatic res;
      res.x = x - obj.x;
      res.y = y - obj.y;
      return res;
    }
    Arithmatic operator/(Arithmatic const &obj)
    {
      Arithmatic res;
      res.x = x / obj.x;
      res.y = y / obj.y;
      return res;
    }
    Arithmatic operator*(Arithmatic const &obj)
    {
      Arithmatic res;
      res.x = x * obj.x;
      res.y = y * obj.y;
      return res;
    }
    void print()
    {
      cout << x << endl;
      cout << y << endl;
    }
  };

  int main()
  {
    Arithmatic a(1, 2), b(2, 3);
    Arithmatic res;
    res = a + b;
    res.print();
    res = a - b;
    res.print();
    res = a / b;
    res.print();
    res = a * b;
    res.print();
    return 0;
  }

```

```
3
5
-1
-1
0
0
2
6
```

MIT World Peace University

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 1

Theory Assignment

Assignment No. 2

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 20

November 28, 2022

1 Questions

1.1 What is the difference between checked and unchecked exceptions in Java? Explain with suitable example.

1.1.1 Checked Exceptions

In broad terms, a checked exception (also called a logical exception) in Java is something that has gone wrong in your code and is potentially recoverable. For example, if there's a client error when calling another API, we could retry from that exception and see if the API is back up and running the second time. A checked exception is caught at compile time so if something throws a checked exception the compiler will enforce that you handle it.

Example:

```java
import java.io.File;
import java.io.FileInputStream;

public class CheckedException {
  public void readFile() {
    String fileName = "file does not exist";
    File file = new File(fileName);
    FileInputStream stream = new FileInputStream(file);
  }
}
```

You simply wrap the Java code which throws the checked exception within a try catch block. This now allows you to process and deal with the exception. With this approach it's very easy to swallow the exception and then carry on like nothing happened. Later in the code when what the method was doing is required you may find yourself with our good friend the NullPointerException.

1.1.2 Unchecked Exceptions

An unchecked exception (also known as an runtime exception) in Java is something that has gone wrong with the program and is unrecoverable. Just because this is not a compile time exception, meaning you do not need to handle it, that does not mean you don't need to be concerned about it.

The most common Java unchecked exception is the good old NullPointerException which is when you are trying to access a variable or object that doesn't exist.

```java
import java.util.ArrayList;
import java.util.List;

public class IndexOutOfBounds {
  public static void main(String[] args) {
    List<String> lst = new ArrayList<>();
    lst.add("item-1");
    lst.add("item-2");
    lst.add("item-3");
    var result = lst.get(lst.size());
  }
}
```

1.1.3 Differncees

- A checked exception is caught at compile time whereas a runtime or unchecked exception is, as it states, at runtime.
- A checked exception must be handled either by re-throwing or with a try catch block, whereas an unchecked isn't required to be handled.
- A runtime exception is a programming error and is fatal whereas a checked exception is an exception condition within your code's logic and can be recovered or re-tried from.

1.2 What is a stream? Explain the different file stream classes with its use?

1.2.1 Stream

A stream is a sequence of data. It is a sequence of bytes. It is a sequence of characters. It is a sequence of objects. It is a sequence of anything. A stream is a sequence of data. It is a sequence of bytes. It is a sequence of characters. It is a sequence of objects. It is a sequence of anything.

In general, a Stream will be an input stream or, an output stream.

- InputStream - This is used to read data from a source.
- OutputStream - This is used to write data to a destination.

1.2.2 FileInputStream

This stream is used for reading data from the files. Objects can be created using the keyword new and there are several types of constructors available.

Once you have InputStream object in hand, then there is a list of helper methods which can be used to read to stream or to do other operations on the stream.

- **public void close()** throws IOException: This method closes the file output stream. Releases any system resources associated with the file. Throws an IOException.
- **protected void finalize()**throws IOException : This method cleans up the connection to the file. Ensures that the close method of this file output stream is called when there are no more references to this stream. Throws an IOException.
- **public int read(int r)throws IOException:** This method reads the specified byte of data from the InputStream. Returns an int. Returns the next byte of data and -1 will be returned if it's the end of the file.
- **public int read(byte[] r) throws IOException:** This method reads r.length bytes from the input stream into an array. Returns the total number of bytes read. If it is the end of the file, -1 will be returned.
- **public int available() throws IOException:** Gives the number of bytes that can be read from this file input stream. Returns an int.

1.2.3 FileOutputStream

FileOutputStream is used to create a file and write data into it. The stream would create a file, if it doesn't already exist, before opening it for output.

Here are two constructors which can be used to create a FileOutputStream object. Once you have OutputStream object in hand, then there is a list of helper methods, which can be used to write to stream or to do other operations on the stream.

- **public void close()** throws IOException: This method closes the file output stream. Releases any system resources associated with the file. Throws an IOException.
- **protected void finalize()throws IOException :** This method cleans up the connection to the file. Ensures that the close method of this file output stream is called when there are no more references to this stream. Throws an IOException.
- **public void write(int w)throws IOException**: This methods writes the specified byte to the output stream.
- **public void write(byte[] w):** Writes w.length bytes from the mentioned byte array to the OutputStream.

Example

```
import java.io.*;
public class fileStreamTest {

  public static void main(String args[]) {

    try {
      byte bWrite [] = {11,21,3,40,5};
      OutputStream os = new FileOutputStream("test.txt");
      for(int x = 0; x < bWrite.length ; x++) {
      os.write( bWrite[x] );   // writes the bytes
      }
      os.close();

      InputStream is = new FileInputStream("test.txt");
      int size = is.available();

      for(int i = 0; i < size; i++) {
      System.out.print((char)is.read() + "  ");
      }
      is.close();
    } catch (IOException e) {
      System.out.print("Exception");
    }
  }
}
```

1.3 What is the difference between the paint() and repaint() methods? Explain with suitable examples.

1.3.1 paint()

The paint() method is called by the AWT when the component needs to be painted. It is called automatically when the component is first shown on the screen. It is also called whenever the contents

of the component need to be updated.

Syntax:

```
public void paint(Graphics g)
```

Parameters: g - the specified Graphics context
Example:

```

import java.awt.*;
public class paint extends Frame {

  public void paint(Graphics g) {
    g.drawString("Welcome", 50, 50);
    g.drawLine(20, 30, 20, 300);
    g.drawRect(70, 100, 30, 30);
    g.fillRect(170, 100, 30, 30);
    g.drawOval(70, 200, 30, 30);
  }

  public static void main(String args[]) {
    paint p = new paint();
    p.setSize(400, 400);
    p.setVisible(true);
  }
}
```

1.3.2 repaint()

The repaint() method is used to force a call to the paint() method. It is used to redraw the component. It is used to update the component.

Syntax:

```
public void repaint()
```

Example:

```
import java.awt.*;
public class repaint extends Frame {

  public void paint(Graphics g) {
    g.drawString("Welcome", 50, 50);
    g.drawLine(20, 30, 20, 300);
    g.drawRect(70, 100, 30, 30);
    g.fillRect(170, 100, 30, 30);
    g.drawOval(70, 200, 30, 30);
  }

  public static void main(String args[]) {
    repaint p = new repaint();
    p.setSize(400, 400);
    p.setVisible(true);
    p.repaint();
  }
}
```

1.4 Write the Java Swing Code that creates a label displaing the text as "Welcome to MITWPU, Pune", with the italics and font size as 15.

```java
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

class Main extends JFrame {
    Main() {
    JLabel lbl = new JLabel("Welcome to MITWPU, Pune");
    lbl.setFont(new Font ("Ariel", Font.ITALIC, 15));
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // setting close operation.
    lbl.setBounds(50, 0, 500, 100);
    setLayout(null); // setting layout using FlowLayout object
    setSize(300, 150); // setting size of Jframe
    add(lbl); // adding Yes button to frame.
    setVisible(true);
  }


  public static void main(String[] args) {
    new Main();
  }
}
```

MIT World Peace University

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 1

Case Study - Elements of an Array

Project Report

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

November 23, 2022

Contents

1 Aim

To perform a Case study on the given problem statement, and implement the problem in C++ and Java

2 Problem Statement

Write a C++ and Java Program to Calculate Average of elements in an Integer Arrays.Take input values.Also display number of elements which are greater than average value.

3 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers : g++ and gcc on linux for C++, and javac, with JDK 18.0.2 for Java

4 Flowchart

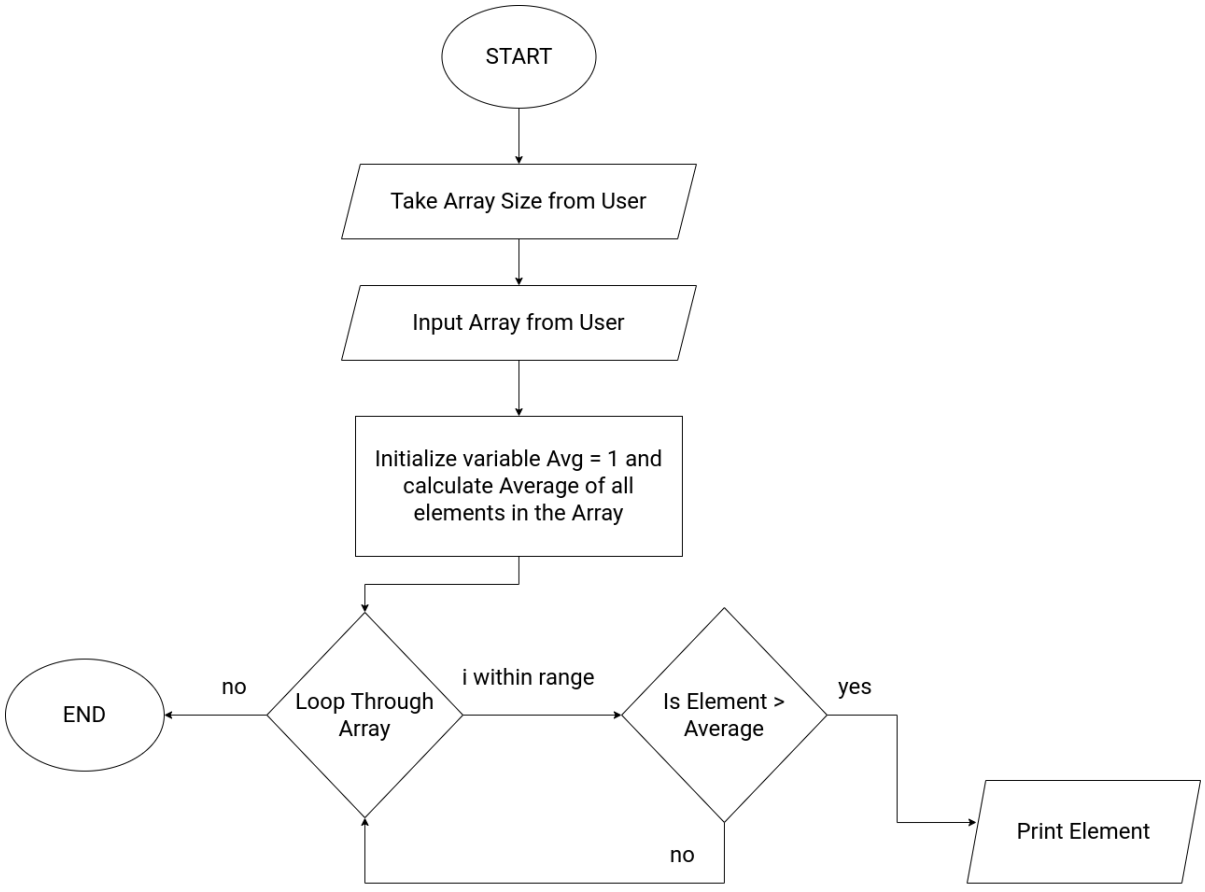


Figure 1: Flowchart for Algorithm

5 Algorithm

```
STEP 1: Start
STEP 2: Input Length of Array from user
STEP 3: Input the Array from the user
STEP 4: Initialize a variable avg to 1, and find average of Array
STEP 5: Loop through the array, if element is greater than average then print it.
STEP 6: Exit.
```

6 Code

6.1 C++ Implementation of Problem

```cpp
// C++ and Java Program to Calculate Average of elements in an Integer Arrays.Take input values.Also display number of elements which are greater than average value.
// Krishnaraj Thadesar
// PA 20 Batch A1

#include <iostream>
using namespace std;
int main()
{
    int size = 10;
    float average = 0;
    cout << "What size array do you want? " << endl;
    cin >> size;
    float arr[size];
    cout << "Enter the elements of the array!" << endl;
    for (int i = 0; i < size; i++)
    {
        cin >> arr[i];
        average += arr[i];
    }
    average /= size;
    cout << "The Average of all the elements in the array is: " << average << endl;
    cout << "The Elements of the Array which are greater than the Average of the Array are: " << endl;
    int count = 0;
    for (int i = 0; i < size; i++)
    {
        if (arr[i] > average)
        {
            cout << arr[i] << endl;
            count++;
        }
    }
    cout << endl << "The Number of Elements greater than the Average are: " << count;

    return 0;
}
```

Listing 1: Main.Cpp

6.2 Java Implementation of Problem

```java
// C++ and Java Program to Calculate Average of elements in an Integer Arrays.Take input values.Also display number of elements which are greater than average value.
// Krishnaraj Thadesar
// PA 20 Batch A1

import java.util.*;;

public class Main {
    static Scanner input = new Scanner(System.in);

    public static void main(String[] args) {
        int size = 10;
        int greater = 0;
        double average = 0;
        System.out.println("Enter the size of the Array that you want to enter");
        size = input.nextInt();
        Double arr[] = new Double[size];
        System.out.println("Enter the Elements of the Array: ");
        for (int i = 0; i < size; i++) {
            arr[i] = input.nextDouble();
            average += arr[i];
        }
        average /= size;
        System.out.println("The Average of All the Elements that you have entered is: " + average);
        System.out.println("The Elements that are above the Average of all the elements are: ");
        for (int i = 0; i < size; i++) {
            if (arr[i] > average) {
                System.out.println(arr[i] + " ");
                greater++;
            }
        }
        System.out.println("The Number of elements greater than the Average is: " + greater);
    }
}
```

Listing 2: Main.java

6.3 Input

1. What are the challenges for multithreading implementation using Java Programming? 2. What is the difference between the start and run method in Java Thread? 3. Which one is better to implement thread in Java? extending Thread class or implementing Runnable? 4. What is the difference between wait and sleep in Java? Explain with example. 5. What is the difference between the submit() and execute() method of Executor and ExecutorService in Java? Explain with example.

1. Length of the Array
2. The Elements of the Array

6.4 Output

6.5 C++ Output

```
What size array do you want?
5
Enter the elements of the array!
1
2
3
4
5
The Average of all the elements in the array is: 3
The Elements of the Array which are greater than the Average of the Array are:
4
5

The Number of Elements greater than the Average are: 2
```

Listing 3: Output for C++

6.6 Java Output

```
Enter the size of the Array that you want to enter
6
Enter the Elements of the Array:
1
4
2
3
6
3
The Average of All the Elements that you have entered is: 3.1666666666666665
The Elements that are above the Average of all the elements are:
4.0
6.0
The Number of elements greater than the Average is: 2
```

Listing 4: Output for Java

MIT World Peace University

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 1

Mini Project with Java - Price Guessing Game
"How Much?"

Project Report

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 20

November 25, 2022

Contents

1 Introduction

This project was made for Submission to Object Oriented Programming with Java and C++ as the End Semester Report. The Motivation behind selecting this topic was that Online shopping has become rather prevelant now a days after COVID, and that has made the average consumer more aware about prices of everyday items, as well as Items out of everyday scope rather well. This game tests that theory, while trying to make it fun and leanring concepts of Java along the way.

The Concept is simple. You are shown a few topics to select from, and then an image along with the title of the Product is shown. There are 4 Choices for its Price which you are supposed to guess within 10 Seconds. For guessing correctly, the time remaining gets added to your score, and you can try again upon guessing incorrectly.

2 Methodology

The Working Methodology of the Game is Discussed below in a few points and elaborated further in the Report.

- There are 2 Active databases Maintained throughout the execution of the Program, MongoDB and CSV. CSV support is added in case the User does not have MongoDB installed in his or her System.
- Upon Starting the Game, it checks for the last time its databse was updated, if it was not within a day, it updates it.
- The databases are updated by quering directly to Amazon and Scraping data. Several Webpages of Amazon and visited, and their pictures and prices are scrapped. They are then stored in the Database.
- The GUI is written entirely in Java Swing and awt.

3 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: IntelliJ Idea Ultimate Edition for Java
Compilers : javac, with JDK 18.0.2 for Java
Database : MongoDb 6.0.3.1

4 Requirements

- **Java 8**
- **Any 32 or 64 bit Operating System**
- **1 GB RAM**
- **Active Internet Connection**

Management

5 Installation and Running

- Navigate to https://github.com/KrishnarajT/How-Much/releases
- Download the .jar file from the releases when it is released that is.
- Navigate there from your terminal and do

```
java -jar ./How_Much.jar
```

6 Database Screenshots

6.1 MongoDB

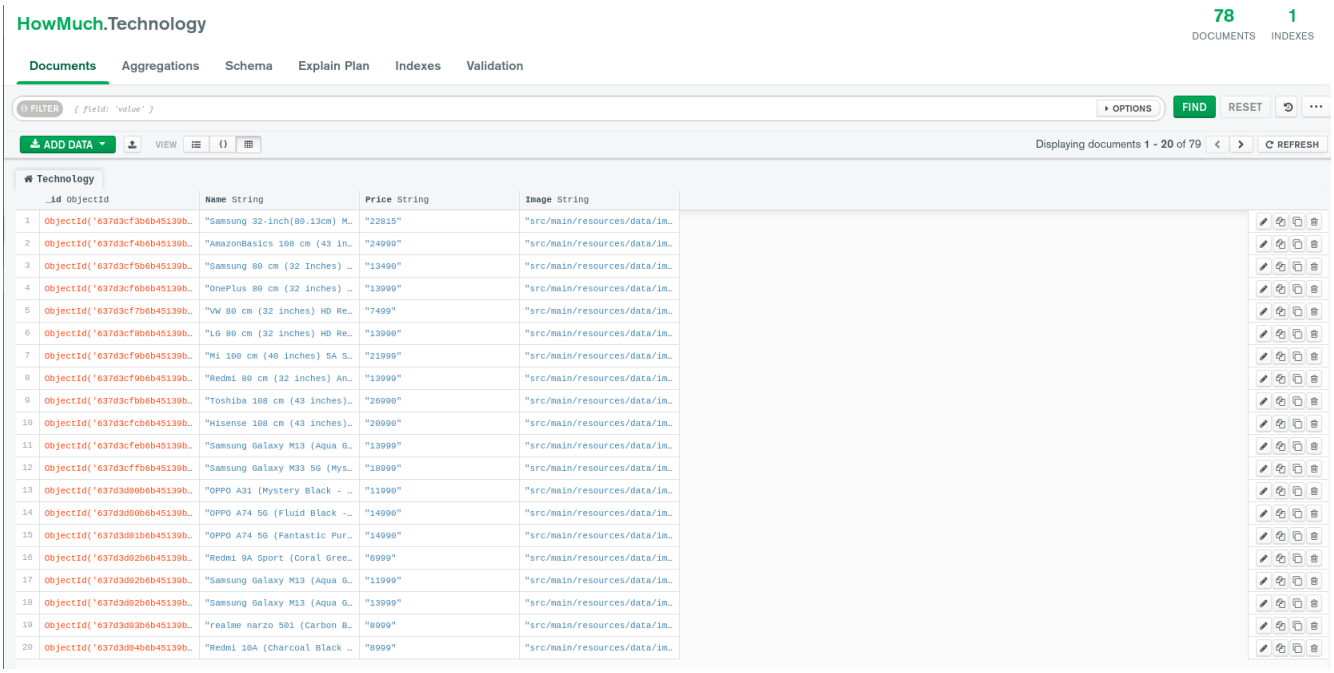


Figure 1: A Screenshot of the MongoDB Compass Showing Records Stored in the Teachnology Schema

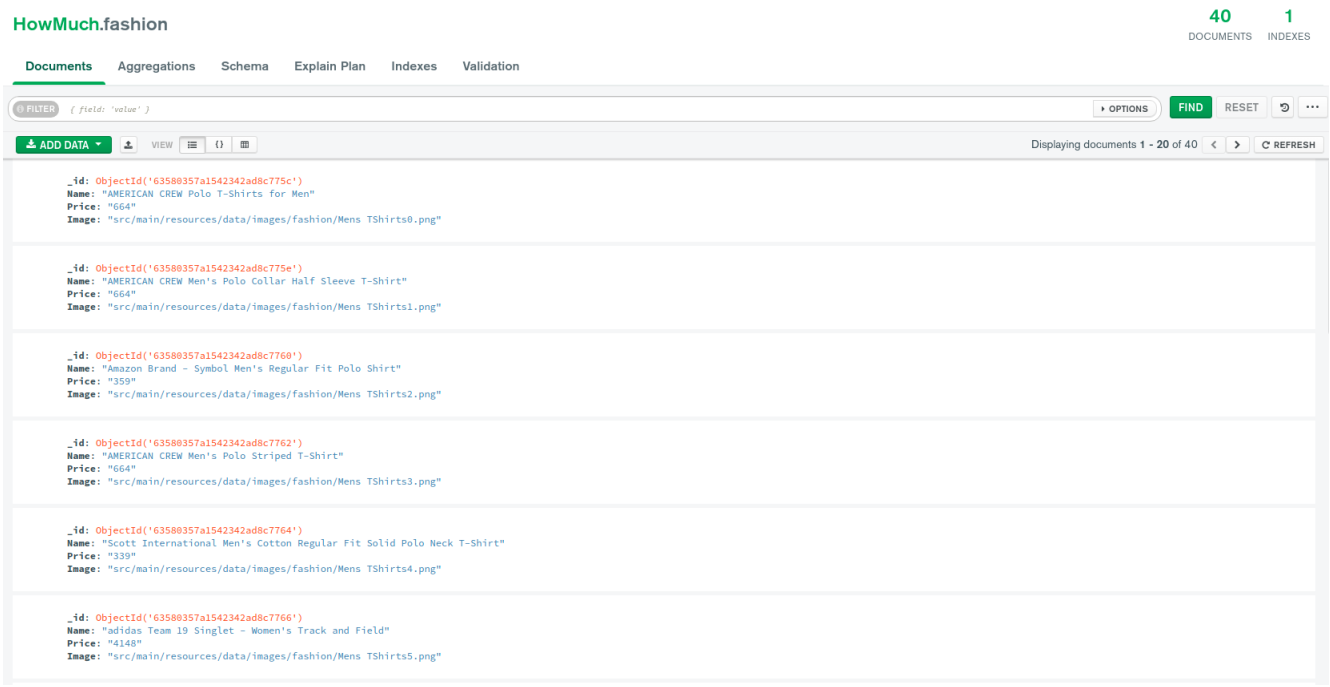


Figure 2: Record Showing the Fashion Schema Documents

6.2 Local CSV Files

	C1	C2	C3
1	EPPE Men's Combo of Round Neck Half Sleeve Dryfit…	630	src/main/resources/data/images/fashion/Mens TShirts0.png
2	Allen Solly Men's Regular Fit T-Shirt	433	src/main/resources/data/images/fashion/Mens TShirts1.png
3	BLUELANDER Printed Round Neck Half Sleeve T-Shirt…	299	src/main/resources/data/images/fashion/Mens TShirts2.png
4	Allen Solly Men's Regular Fit T-Shirt	539	src/main/resources/data/images/fashion/Mens TShirts3.png
5	Allen Solly Men Polo	758	src/main/resources/data/images/fashion/Mens TShirts4.png
6	U.S. POLO ASSN. Men T-Shirt	352	src/main/resources/data/images/fashion/Mens TShirts5.png
7	Amazon Brand - Symbol Men T-Shirt	919	src/main/resources/data/images/fashion/Mens TShirts6.png
8	AELOMART Men's T Shirt	497	src/main/resources/data/images/fashion/Mens TShirts7.png
9	Scott International Men's Regular Fit T-Shirt (Pa…	474	src/main/resources/data/images/fashion/Mens TShirts8.png
10	Amazon Brand - Symbol Men's Regular T-Shirt	859	src/main/resources/data/images/fashion/Mens TShirts9.png
11	Park Avenue Full Sleeve Shawl Collar Dark Brown S…	4399	src/main/resources/data/images/fashion/Formal Suits0.png
12	Park Avenue Solid Rayon Blend Dark Blue Regular F…	4599	src/main/resources/data/images/fashion/Formal Suits1.png
13	Park Avenue Dark Grey Suits	4499	src/main/resources/data/images/fashion/Formal Suits2.png
14	Park Avenue Medium Grey Suits	5849	src/main/resources/data/images/fashion/Formal Suits3.png
15	Razab Enterprises (SAAYA) 5 Button Bandhgala/Jodh…	3045	src/main/resources/data/images/fashion/Formal Suits4.png
16	Arrow Men's Polyester Blend Formal Business Suit …	4979	src/main/resources/data/images/fashion/Formal Suits5.png

Figure 3: Screenshot of the Local CSV File

7 Unique Features

7.1 Dark Mode

Dark mode is toggled by a switch. It simply flips a boolean variable statically defined in Colors.java. Other classes will then set Colors on their screens depending on this variable, for each Swing element in their Panel or Frame.

Figure 4: Dark mode Turned on

Figure 5: Dark mode Turned Off

7.2 Data Backup

Data backup is an important feature that ensures the user never has to face a situation where there is no product to be loaded on the Screen.

- There are 3 Databases maintained.
- When updating, the program updates MongoDB and CSV if they have not been updated.
- Each of the database have their own text file to maintain when the last time it was that they got updated.
- They are updated only once a day, as updating takes time.
- Updating is done in separate threads running in the Background.
- There is a 3rd backup database of CSV files that just duplicates the current state of the CSV database each time the user exits the program.
- The Game can update the Database in the Background while the user is playing the game, and at this point the backup database can be used.

7.3 Web Scrapping

Figure 6: Product on Amazon

- Every Webpage on amazon with some product has products that look like the one above.

- The HTML page is scrapped, and the respective divs are searched in it to find each product and its price.
- It is then stored in the database after downloading and parsing the HTML file.

7.4 Working Login and Account Creation

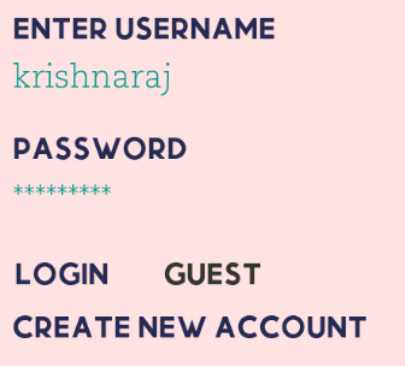


Figure 7:

- All the Requirements of a simple login are satisfied here.
- After the user inputs the username, it is valided in the local CSV file.
- If found, the password is expected, checked and login is permitted.
- If not found, password is validated, and a new user account creation is permitted.

8 Color Schemes Used

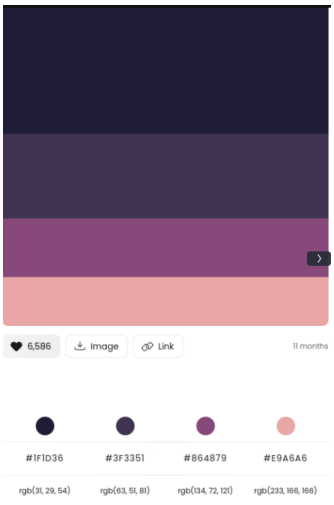


Figure 8: Color Palette

The Above Colors where used and are defined in the Colors.java.

9 Screenshots of the Project

9.1 The Login Page

Figure 9: The Login page after a successful login

9.2 The Menu Screen

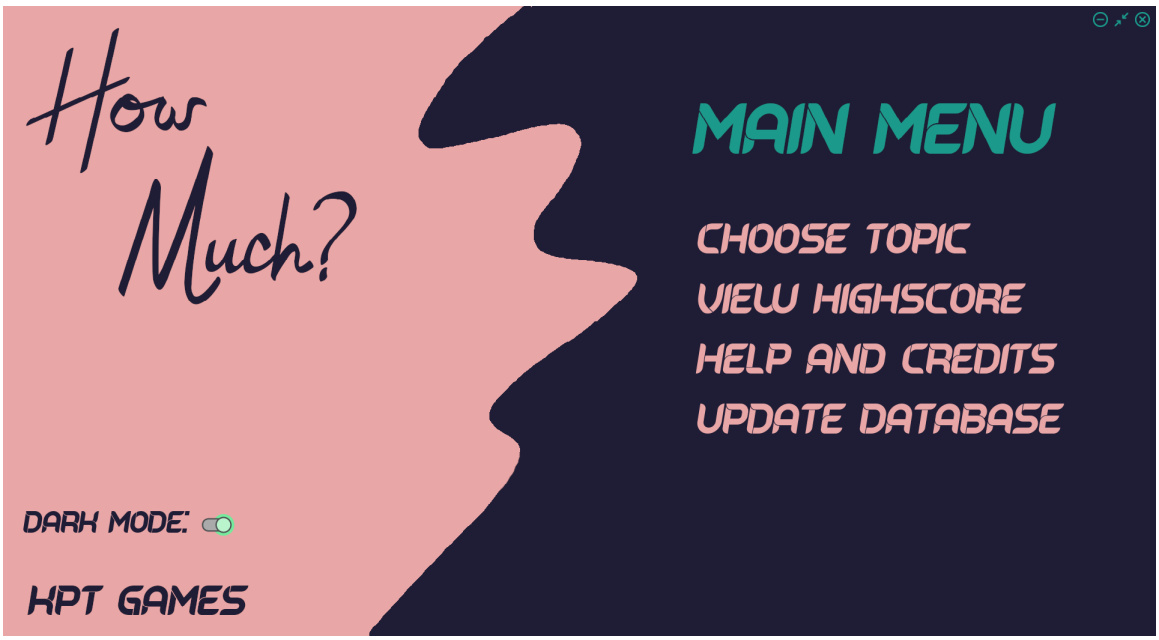


Figure 10:

9.3 The Topic Selection Screen

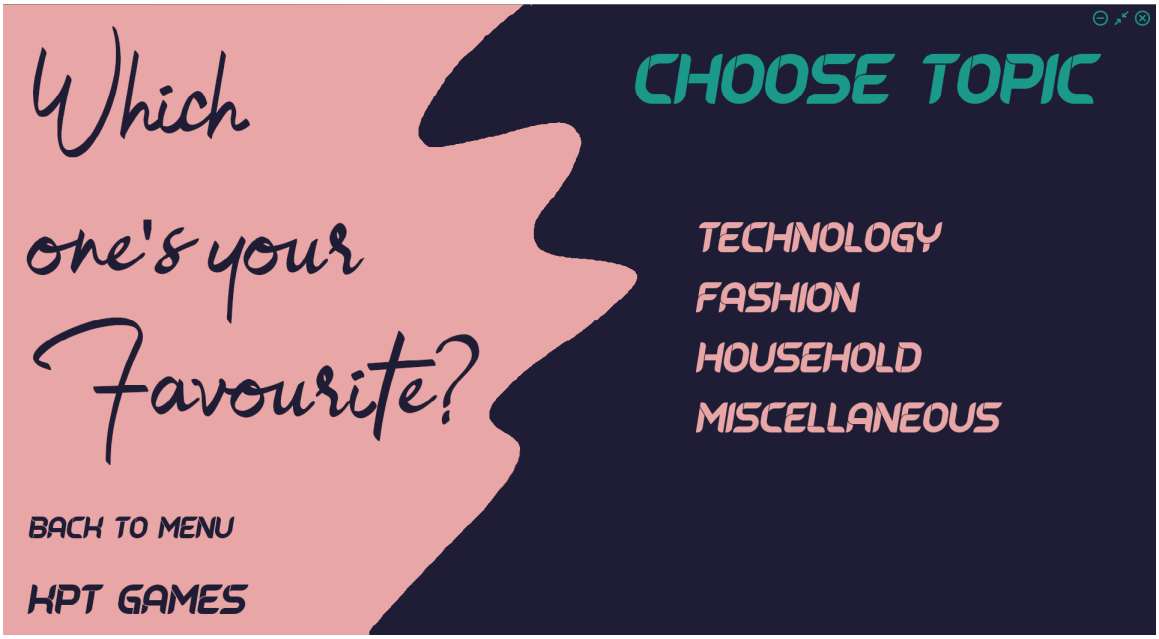


Figure 11: The Login page after a successful login

9.4 The Highscore Screen

Figure 12: The Login page after a successful login

9.5 The Help and About

Figure 13: The Login page after a successful login

9.6 The Game Over Screen

Figure 14: The Login page after a successful login

10 Walk-Through of the Files

10.1 Project Structure

Figure 15:

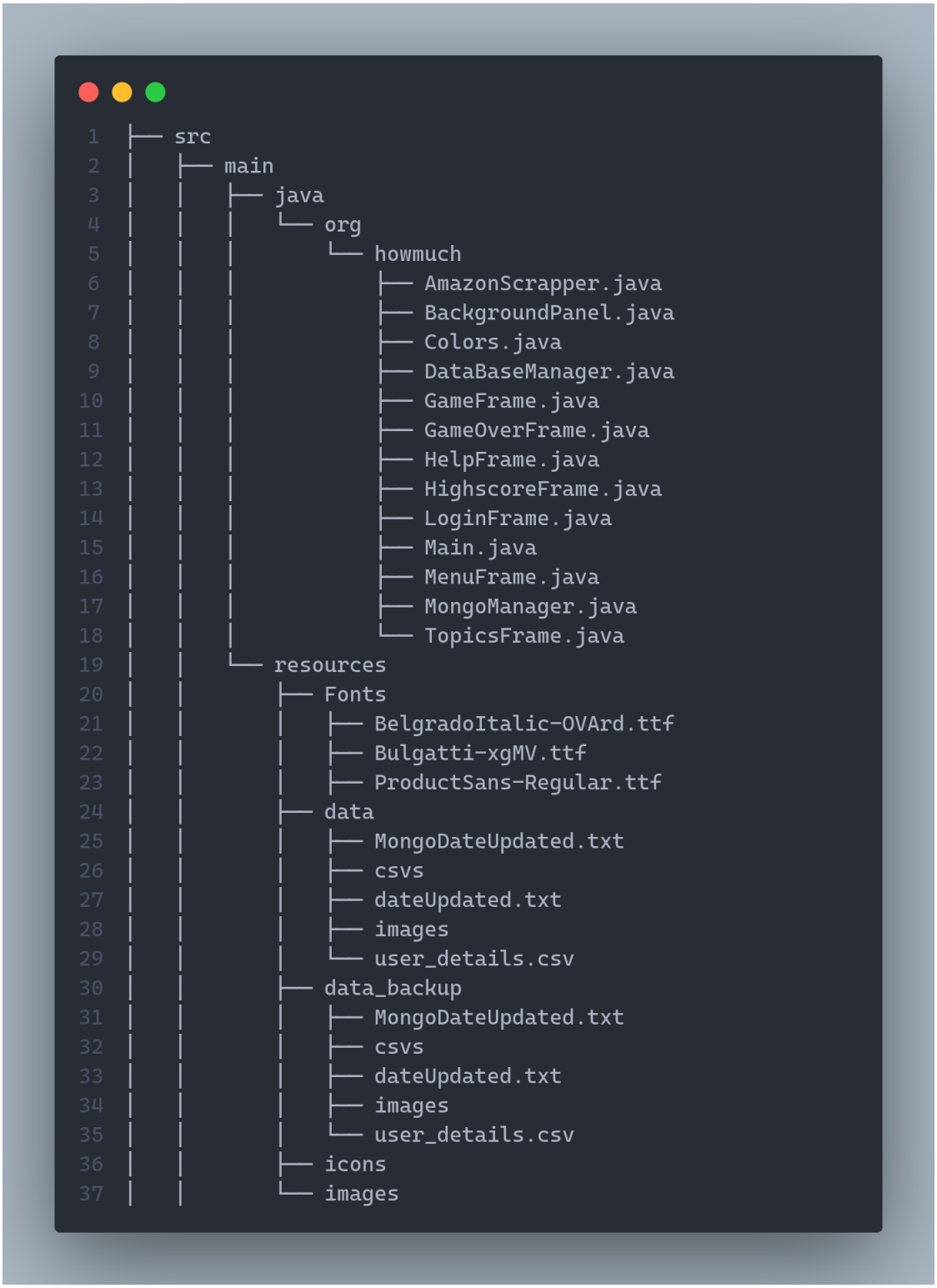

src
main
java
org
howmuch
AmazonScrapper.java
BackgroundPanel.java
Colors.java
DataBaseManager.java
GameFrame.java
GameOverFrame.java
HelpFrame.java
HighscoreFrame.java
LoginFrame.java
Main.java
MenuFrame.java
MongoManager.java
TopicsFrame.java
resources
Fonts
BelgradoItalic-OVArd.ttf
Bulgatti-xgMV.ttf
ProductSans-Regular.ttf
data
MongoDateUpdated.txt
csvs
dateUpdated.txt
images
user_details.csv
data_backup
MongoDateUpdated.txt
csvs
dateUpdated.txt
images
user_details.csv
icons
images

Figure 16:

10.2 TopicsFrame.java

This file manages the entire topic selection screen. It has various functions regarding showing the topics on the screen. It then sets static variables defined in the Main class with respect to the selected Topic.

10.3 MongoManager.java

Important Class, which manages all interactions with Mongodb. It establishes connection with it, and flips a statically defined variable in Main called usingMongo to true or false depending on the success of the connection. It also has functions to update, clear, and retrieve values to and from the Mongo Database.

10.4 MenuFrame.java

This file manages the entire Menu selection screen. It has various functions regarding showing the topics on the screen. It then sets static variables defined in the Main class with respect to the selected Topic. It also has the Dark mode toggle, which flips a boolean called usingDarkMode defined in Colors.java.

10.5 Main.java

This class calls all the other classes. It also has the main function. It uses multithreading to update the database at the same time as displaying the GUI. It has a function that manages the interactions between all the other classes. It also has several statically defined variables.

10.6 LoginFrame.java

A Class that manages everything defined in the Login Screen. It has functions to check if the password fits the given criteria, and it queries and updates the database using functions defined in other classes.

10.7 HighscoreFrame.java

A Simple class that just displays the High scores of the User. It retrieves that data using database functions, and shows the top 5 Highscoring Users along with their scores.

10.8 HelpFrame.java

A Simple Class that simply displays what to do in the game, how to play and the credits.

10.9 GameOverFrame.java

A Simple class that just shows the score and gives an option to the user to go back to the main menu to try again if the game was lost, or won.

10.10 GameFrame.java

This is the Main class, in that it shows the actual game. It has functions to check if the databases are working properly, and what to refer in case some of them dont work. It retrieves data using functions defind in other classes. Calculates 4 suitable options depending on the Correct price, and displayes everything on the Screen.

10.11 DataBaseManager.java

Important class that manages the Local CSV files. It updates, retrieves, and clears it. It also has functions to check the database for login functions, like password matching, username matching, adding username etc.

10.12 Colors.java

Another important class that has all the colors defined in it. It has a function to reassign colors, which is called every time the Dark Mode switch is flipped.

10.13 BackgroundPanel.java

Important class, as it is the panel that is used by all the other screens to display the background. It has a function to set the background as the Swing JPanel background by taking arguement of the location of the image to be inserted, while maintaining aspect ratio of the image.

10.14 AmazonScrapper.java

A very important class, as it has functions to actually scrap the data from amazon, and parse it. It then verifies the data, checks for invalid characters, and if everything is fine it calls functions from teh database class to insert new data into the databases.

11 Conclusion and Topics Learnt

A lot of topics were learnt in the process of making this Project. It was very useful to make this Game, and it got me a lot more fluent in writing Java code.

1. Multithreading was Understood in a greater depth, and implemented many times.
2. Web Scrapping was Learnt and in the process various java Libraries were used and understood.
3. Swing in java was learnt in a higher detail.
4. Database Management was understood.
5. JDBC Drivers, Connections of java with MongoDB and MySQL were learnt in detail.
6. Several Bugs were Resolved and as a Result of that programming skills were improved.
7. Designing skills were also improved.

12 Dependencies

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache
    .org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>How-Much</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>18</maven.compiler.source>
        <maven.compiler.target>18</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <repositories>
        <repository>
            <id>groupdocs-artifacts-repository</id>
            <name>GroupDocs Artifacts Repository</name>
            <url>https://releases.groupdocs.com/java/repo/</url>
        </repository>
    </repositories>

    <dependencies>
        <dependency>

            <groupId>net.sourceforge.htmlunit</groupId>
            <artifactId>htmlunit</artifactId>
            <version>2.60.0</version>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>2.13.2.1</version>
        </dependency>

        <dependency>
            <groupId>org.mongodb</groupId>
            <artifactId>mongo-java-driver</artifactId>
            <version>3.12.2</version>
        </dependency>

        <dependency>
            <groupId>com.opencsv</groupId>
            <artifactId>opencsv</artifactId>
            <version>4.1</version>
        </dependency>

        <dependency>
            <groupId>com.groupdocs</groupId>
            <artifactId>groupdocs-conversion</artifactId>
            <version>22.8.1</version>
        </dependency>

```

```
    </dependencies>

</project>
```

Listing 1: Maven Dependency File

13 References

- https://www.notion.so/045f2a28abfe4a82a3ba0e8251c0e196?v=1ddbea2937324f79af4eae827f8a3132 chrome://newtab/
- https://classroom.google.com/u/0/w/NTUxMDUyMzM4MDk4/t/all
- https://webscraping.pro/scraping-amazon-webdriver-java/
- https://www.geeksforgeeks.org/scraping-amazon-product-information-using-beautiful-soup/
- https://www.scrapingbee.com/blog/web-scraping-amazon/
- https://docs.oracle.com/cd/E50453_01/doc.80/e50452/run_java_guis.htm#:~:text=In%20Java%20applications%2C%20the%20components,GUI%20forms%20can%20be%20built.
- https://www.javatpoint.com/multithreading-in-java
- https://www.digitalocean.com/community/tutorials/multithreading-in-java
- https://www.guru99.com/multithreading-java.html
- https://one.trustedstream.life/space-robot/?pl=uy-e9pFAkEqFCifo9sDBZw&sm=space-robot&hash=6I1UsQZL8I67jbxjdCa5gg&exp=1669365585
- https://www.mongodb.com/languages/java
- https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html#:~:text=The%20Action%20interface%20provides%20a,be%20accessed%20by%20several%20controls.&text=Defines%20the%20data%20model%20used,Slider%20s%20and%20ProgressBar%20s.
- Credits to Teachers and Friends for their constant help and support.

14 Code Files

```java
package org.howmuch;

import java.awt.*;

public class Colors {

    static Boolean DarkMode = false;

    // Light mode Colors
    static Color light_bgColor = new Color(255, 227, 227);
    static Color light_primaryColor = new Color(38, 42, 83);
    static Color light_secondaryColor = new Color(255, 160, 160);
    static Color light_accentColor = new Color(27, 153, 139);

```

```
    // Dark Mode Colors
    static Color dark_bg_color = new Color(31, 29, 54);
    static Color dark_primaryColor = new Color(233, 166, 166);
    static Color dark_secondaryColor = new Color(175, 89, 159);
    static Color dark_accentColor = new Color(27, 153, 139);

    static Color bgColor = DarkMode ? dark_bg_color : light_bgColor;
    static Color primaryColor = DarkMode ? dark_primaryColor : light_primaryColor;
    static Color secondaryColor = DarkMode ? dark_secondaryColor :
    light_secondaryColor;
    static Color accentColor = DarkMode ? dark_accentColor : light_accentColor;

    public static void reassignColors() {
        bgColor = DarkMode ? dark_bg_color : light_bgColor;
        primaryColor = DarkMode ? dark_primaryColor : light_primaryColor;
        secondaryColor = DarkMode ? dark_secondaryColor : light_secondaryColor;
        accentColor = DarkMode ? dark_accentColor : light_accentColor;
    }
}
```

Listing 2: Main Java File

```
package org.howmuch;

import javax.swing.*;
import java.awt.*;

public class BackgroundPanel extends JPanel {
    private Image background;

    public void paintComponent(Graphics g) {

        super.paintComponent(g);
        int width = this.getSize().width;
        int height = this.getSize().height;

        if (this.background != null) {
            // Add the size of the window in drawImage method()
            g.drawImage(this.background, 0, 0, width, height, null);
        }
    }

    public void setBackground(String imagePath) {
        // Simply sets the background as the one that you have provided. It needs
    to be
        // a png file (I think)
        this.background = new ImageIcon(imagePath).getImage();
        repaint();
    }

}
```

Listing 3: Main Java file

```
package org.howmuch;

import org.xml.sax.SAXException;
import javax.swing.*;
import javax.xml.parsers.ParserConfigurationException;
```

```
6  import java.awt.*;
7  import java.io.*;
8  import java.time.LocalDate;
9
10 // You have to extend the thread class to create a new thread for running the
       databases.
11 public class Main extends Thread {
12
13     // Statically defining important variables used throughout the game. They are
14     // statically defined coz they are used by other classes very often.
15     public static String[] Topics = new String[] { "Technology", "Fashion", "
       Household", "Miscellaneous" };
16     public static String currentTopic = Topics[0];
17     static final int WIDTH = 1280, HEIGHT = 720;
18     static boolean maximized = false, isGuest = true, grantAccess = false,
       isLocalDatabaseUpToDate = false,
19             isMongoUpToDate = false, usingMongo = false;
20
21     // Declaring Objects of other classes that we are going to call from main.
22     static LoginFrame loginFrame;
23     static MenuFrame menuFrame;
24     static HelpFrame helpFrame;
25     static HighscoreFrame highscoreFrame;
26     static TopicsFrame topicsFrame;
27     static GameFrame gameFrame;
28     static GameOverFrame gameOverFrame;
29
30     static Font buttonFont, textFont, password_font, options_font, emoji_font;
31     static JButton exit_btn, resize_btn, minimize_btn;
32     static JPanel basicButtons_pnl;
33
34     // These are the icons from where we get the resize, exit and the minimize
35     // button. They are custom made coz they look better,
36     // eliminate the need for the titlebar making the UI look cleaner, albeit less
37     // useful.
38     // They also let you have full control over what you want to do when they are
39     // pressed, and what you wanna call, which you cant do without them.
40     // You can also control now exactly the resizing behaviour of your software.
41     static ImageIcon exit = new ImageIcon("src/main/resources/icons/circle_delete.
       png");
42     static Image exit_image = exit.getImage().getScaledInstance(25, 25, Image.
       SCALE_SMOOTH);
43     static ImageIcon minimize = new ImageIcon("src/main/resources/icons/
       circle_minus.png");
44     static Image minimize_image = minimize.getImage().getScaledInstance(25, 25,
       Image.SCALE_SMOOTH);
45     static ImageIcon resizeUp = new ImageIcon("src/main/resources/icons/resize_3.
       png");
46     static Image resizeUp_image = resizeUp.getImage().getScaledInstance(25, 25,
       Image.SCALE_SMOOTH);
47     static ImageIcon resizeDown = new ImageIcon("src/main/resources/icons/resize_4
       .png");
48     static Image resizeDown_image = resizeDown.getImage().getScaledInstance(25,
       25, Image.SCALE_SMOOTH);
49
50     /**
51      * Creates fonts by instantiating the font objects with their respective fonts
52      * stored locally. Static and used everywhere. Its an important function and
53      * gets called in almost every class constructor.
```

```
54      */
55     public static void createFonts() {
56         try {
57             GraphicsEnvironment ge = GraphicsEnvironment.
    getLocalGraphicsEnvironment();
58 
59             // Used for Buttons Almost everywhere.
60             buttonFont = Font
61                     .createFont(Font.TRUETYPE_FONT, new File("/run/media/
    krishnaraj/Programs/Java/How Much/src/main/resources/Fonts/BelgradoItalic-OVArd
    .ttf"))
62                     .deriveFont(50f);
63             // Used Mostly on the Login Page.
64             textFont = Font.createFont(Font.TRUETYPE_FONT, new File("/run/media/
    krishnaraj/Programs/Java/How Much/src/main/resources/Fonts/MomcakeBold-WyonA.
    otf"))
65                     .deriveFont(50f);
66             // Used for password Entering
67             password_font = Font
68                     .createFont(Font.TRUETYPE_FONT, new File("/run/media/
    krishnaraj/Programs/Java/How Much/src/main/resources/Fonts/CaeciliaLTPro45Light
    .TTF"))
69                     .deriveFont(35f);
70             // Used only for Emojis
71             emoji_font = Font.createFont(Font.TRUETYPE_FONT,
72                     new File("/run/media/krishnaraj/Programs/Java/How Much/src/
    main/resources/Fonts/NotoEmoji-VariableFont_wght.ttf")).deriveFont(35f);
73             // Used to show the Price, needs to contain the Rupee symbol
74             options_font = Font
75                     .createFont(Font.TRUETYPE_FONT, new File("/run/media/
    krishnaraj/Programs/Java/How Much/src/main/resources/Fonts/ProductSans-Regular.
    ttf"))
76                     .deriveFont(35f);
77 
78             // registering them locally, not required.
79             ge.registerFont(textFont);
80             ge.registerFont(buttonFont);
81             ge.registerFont(password_font);
82             ge.registerFont(emoji_font);
83             ge.registerFont(options_font);
84 
85         } catch (FontFormatException | IOException e) {
86             e.printStackTrace();
87             System.out.println("Couldnt create the fonts. ");
88         }
89     }
90 
91     /*
92      * Function to Create the resize, minimize and the exit button, they are all
93      * placed in a panel, so that you can move them around easily without the
    hassle
94      * of moving each thing. Just move the panel. Here we define them.
95      */
96     public static void createBasicButtonPanel() {
97         basicButtons_pnl = new JPanel();
98         FlowLayout fl = new FlowLayout(FlowLayout.LEFT, 10, 0);
99         basicButtons_pnl.setLayout(fl);
100 
101         exit_btn = new JButton();
```

```
102         exit_btn.setIcon(new ImageIcon(exit_image));
103         exit_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
104         exit_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
105         exit_btn.setBounds(new Rectangle(25, 25));
106         exit_btn.setFont(buttonFont.deriveFont(44f));
107         exit_btn.setFocusPainted(false);
108         exit_btn.setContentAreaFilled(false);
109         exit_btn.setOpaque(true);
110         exit_btn.setBorder(null);
111 
112         resize_btn = new JButton();
113         if (Main.maximized) {
114             resize_btn.setIcon(new ImageIcon(resizeDown_image));
115         } else {
116             resize_btn.setIcon(new ImageIcon(resizeUp_image));
117         }
118         resize_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
119         resize_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
120         resize_btn.setBounds(new Rectangle(25, 25));
121         resize_btn.setFont(buttonFont.deriveFont(44f));
122         resize_btn.setFocusPainted(false);
123         resize_btn.setContentAreaFilled(false);
124         resize_btn.setOpaque(true);
125         resize_btn.setBorder(null);
126 
127         minimize_btn = new JButton();
128         minimize_btn.setIcon(new ImageIcon(minimize_image));
129         minimize_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
130         minimize_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
131         minimize_btn.setBounds(new Rectangle(25, 25));
132         minimize_btn.setFont(buttonFont.deriveFont(44f));
133         minimize_btn.setFocusPainted(false);
134         minimize_btn.setContentAreaFilled(false);
135         minimize_btn.setOpaque(true);
136         minimize_btn.setBorder(null);
137 
138         // Adding them to the panel here.
139         basicButtons_pnl.add(minimize_btn);
140         basicButtons_pnl.add(resize_btn);
141         basicButtons_pnl.add(exit_btn);
142     }
143 
144     /**
145      * @param status = 1: Call Main Menu <br>
146      *               status = 2: Call Topic Selection<br>
147      *               status = 3: Call Help and Credits<br>
148      *               status = 4: View Highscores<br>
149      *               status = 5: Update Database<br>
150      *               status = 6: Start Game<br>
151      *               status = 7: Game over Screen<br>
152      *               status = 0: Exit Game<br>
153      *
154      *               Important Function as it decides to change to another frame,
155      *               provides some security with grantedAccess boolean,
156      *               and Also does the mandatory things that need to be done if
    the
157      *               close button is pressed.
158      **/
159     public static void changeFrame(int status) {
```

```
160         // Status is 0 when you wanna quit, so we gotta do some stuff before you
    quit,
161         // like creating the backup.
162         if (status == 0) {
163 
164             // Create a local backup of the users file irrespective of what was
    done during
165             // gameplay.
166             DataBaseManager.createLocalDatabaseBackupOfUsers();
167 
168             // If the user is a guest, then the index is less than 0, in which
    case dont
169             // update anything.
170             if (DataBaseManager.USER_INDEX < 0) {
171                 System.out.println("You are a guest, so not updating anything. \n"
    );
172             } else {
173                 // If its a user, then update the user score. The index is known
    already in a
174                 // static variable.
175                 DataBaseManager.updateUserScore();
176             }
177 
178             // This is what keeps track of when the last time was that the
    database was
179             // updated. You dont need to update it every time you run the game.
180             String lastUpdateDate = "";
181 
182             // Opening the Backup Date file and checking the last time it was
    backed up.
183             File dateFile = new File(DataBaseManager.LOCAL_BACKUP_DATEFILE);
184             if (dateFile.exists()) {
185                 try (BufferedReader br = new BufferedReader(new FileReader(
    dateFile))) {
186                     lastUpdateDate = br.readLine();
187 
188                     // If the database was backed up last today, then dont do it.
189                     if (lastUpdateDate.equals(String.valueOf(LocalDate.now()))) {
190                         System.out.println("Backup DataBases are Up to Date!");
191                     } else {
192                         // Else update it.
193                         DataBaseManager.createLocalDatabaseBackup();
194                     }
195                 } catch (IOException e) {
196                     throw new RuntimeException(e);
197                 } catch (NullPointerException exception) {
198                     System.out.println("Nothing in the Date File. ");
199                 }
200             } else {
201                 // If the file itself doesnt exist, then clearly there doesnt
    exist any backup,
202                 // so we better back up at that point.
203                 try {
204                     DataBaseManager.createLocalDatabaseBackup();
205                 } catch (Exception e) {
206                     System.out.println("You havent really created the database yet
    , so not creating backup either. ");
207                 }
208             }
```

```
209             // Exit game
210             System.out.println("Thanks for Playing! ");
211             System.exit(0);
212         }
213         if (grantAccess) {
214             System.out.println("Access Granted!");
215             switch (status) {
216                 case 1 -> {
217                     // Showing Main Menu
218                     grantAccess = false;
219                     menuFrame = new MenuFrame();
220                 }
221                 case 2 -> {
222                     // Showing the TopicsFrame
223                     grantAccess = false;
224                     topicsFrame = new TopicsFrame();
225                 }
226                 case 3 -> {
227                     // Showing the Help Screen
228                     grantAccess = false;
229                     helpFrame = new HelpFrame();
230                 }
231                 case 4 -> {
232                     // Showing Highscores
233                     grantAccess = false;
234                     highscoreFrame = new HighscoreFrame();
235                 }
236                 case 5 -> {
237                     System.out.println("Updating Database");
238 
239                     // Instead of overwriting the files, or appending to them, as
    they contain old
240                     // data,
241                     // we will just erase them altogether and create them again.
242                     DataBaseManager.clearLocalDatabase();
243                     MongoManager.clearMongoDb();
244 
245                     // Scrap everything and Start Saving
246                     AmazonScrapper obj = new AmazonScrapper();
247                     try {
248                         AmazonScrapper.scrapAndSave();
249                     } catch (Exception e) {
250                         System.out.println("Couldnt update the database, there was
     some problem. It was");
251                         System.out.println(e.getMessage());
252                     }
253                     // Just copy everything to the backup either way.
254                     DataBaseManager.createLocalDatabaseBackup();
255 
256                     File dateFile;
257 
258                     // Updating the Mongo and Local Database File.
259                     dateFile = new File(DataBaseManager.LOCAL_DATEFILE);
260                     try (FileWriter f = new FileWriter(dateFile, false)) {
261                         f.write(String.valueOf(LocalDate.now()));
262                     } catch (IOException e) {
263                         throw new RuntimeException(e);
264                     }
265                     dateFile = new File(DataBaseManager.LOCAL_MONGODATEFILE);
```

```
266                     try (FileWriter f = new FileWriter(dateFile, false)) {
267                         f.write(String.valueOf(LocalDate.now()));
268                     } catch (IOException e) {
269                         throw new RuntimeException(e);
270                     }
271                 }
272                 case 6 -> {
273                     // Showing Game Screen
274                     grantAccess = false;
275                     gameFrame = new GameFrame();
276                 }
277                 case 7 -> {
278                     // This is only called by the gameframe, which has a timer,
    which is what calls
279                     // this function, and as its in a different class,
280                     // you have to close the things from here coz that timer cant
    access its parent
281                     // class properties.
282                     gameFrame.setVisible(false);
283                     gameFrame.dispose();
284 
285                     // Show GameOverScreen
286                     gameOverFrame = new GameOverFrame();
287                 }
288                 default -> {
289                     // In Case something goes really wrong, just backup and exit.
290                     DataBaseManager.createLocalDatabaseBackup();
291 
292                     // Exit game
293                     System.out.println("Thanks for Playing! ");
294                     System.exit(0);
295                 }
296             }
297         } else {
298             System.out.println("Access Denied Who are you? What are you trynna do
    here? ");
299             System.exit(0);
300         }
301     }
302 
303     /*
304      * This function is overridden from the Thread class, coz its empty there, and
305      * thread.start calls this function.
306      * And this is where you put loops or something in case you wanna do something
307      * for ever as a game Loop and access data members stored somewhere else and
308      * written to by some other classes.
309      * The Job of this function here is important in that its the first function
310      * that is real multithread. It checks the database, and if they are not up to
311      * date, it updates them.
312      */
313     public void run() {
314 
315         // Just establish the connection, and if thats not possible, then we
    clearly
316         // arent gonna be using mongo.
317         usingMongo = MongoManager.establishConnectionWithMongo();
318 
319         // Same logic as demod in changeFrame()
320         String lastUpdateDate = "";
```

```
321
322         // Checking the Local CSV Files
323         File dateFile = new File(DataBaseManager.LOCAL_DATEFILE);
324         if (dateFile.exists()) {
325             try (BufferedReader br = new BufferedReader(new FileReader(dateFile)))
     {
326                 lastUpdateDate = br.readLine();
327                 System.out.println(lastUpdateDate);
328                 if (lastUpdateDate.equals(String.valueOf(LocalDate.now()))) {
329                     System.out.println("Local DataBases are Up to Date!");
330                     isLocalDatabaseUpToDate = true;
331                 }
332             } catch (IOException e) {
333                 throw new RuntimeException(e);
334             } catch (NullPointerException exception) {
335                 System.out.println("Nothing in the Local Date File. ");
336             }
337         }
338
339         // Now check the mongodb database date file to check when was the last
    time it
340         // was updated. Same Logic tho.
341         dateFile = new File(DataBaseManager.LOCAL_MONGODATEFILE);
342         if (dateFile.exists()) {
343             try (BufferedReader br = new BufferedReader(new FileReader(dateFile)))
     {
344                 lastUpdateDate = br.readLine();
345                 System.out.println(lastUpdateDate);
346                 if (lastUpdateDate.equals(String.valueOf(LocalDate.now()))) {
347                     System.out.println("Mongo DataBases are Up to Date!");
348                     isMongoUpToDate = true;
349                 }
350             } catch (IOException e) {
351                 throw new RuntimeException(e);
352             } catch (NullPointerException exception) {
353                 System.out.println("Nothing in the mongo Date File. ");
354             }
355         }
356
357         // If say one of them is not updated, then we gotta scrap amazon.
358         if (!isLocalDatabaseUpToDate || (usingMongo && !isMongoUpToDate)) {
359
360             System.out.println("Beginning to Scrap Data From Amazon, as one of the
     DataBases isnt updated. ");
361             if (!isLocalDatabaseUpToDate) {
362                 // As an edge case, if mongo isnt up to date, we dont wanna clear
    the local one.
363                 DataBaseManager.clearLocalDatabase();
364             }
365             if (usingMongo && !isMongoUpToDate) {
366                 // If the local one isnt up to date we dont wanna clear mongo.
367                 MongoManager.clearMongoDb();
368             }
369
370             // Scrap and save, as at this point we already know what works and
    what doesnt,
371             // and what is updated and what isnt,
372             // AmazonScrapper class can figure out where to save stuff. After that
373             // everything would have to be updated.
```

```
374             AmazonScrapper obj = new AmazonScrapper();
375             try {
376                 AmazonScrapper.scrapAndSave();
377                 isLocalDatabaseUpToDate = true;
378 
379                 // writing to the date file coz we must have updated at this point
380                 dateFile = new File(DataBaseManager.LOCAL_DATEFILE);
381                 try (FileWriter f = new FileWriter(dateFile, false)) {
382                     f.write(String.valueOf(LocalDate.now()));
383                 } catch (IOException e) {
384                     throw new RuntimeException(e);
385                 }
386                 System.out.println("Updated the local database, no need to depend
    on the backup anymore");
387 
388                 if (usingMongo) {
389                     // Coz at this point it has to be, as we just scrapped and
    didnt get any erros.
390                     isMongoUpToDate = true;
391 
392                     // writing to the date file coz we must have updated at this
    point
393                     dateFile = new File(DataBaseManager.LOCAL_MONGODATEFILE);
394                     try (FileWriter f = new FileWriter(dateFile, false)) {
395                         f.write(String.valueOf(LocalDate.now()));
396                     } catch (IOException e) {
397                         throw new RuntimeException(e);
398                     }
399                     System.out.println("Updated the Mongo database, no need to
    depend on the local one anymore");
400                 }
401             } catch (Exception e) {
402                 System.out.print("Couldnt update one of the databases, in the case
     that one of them wasnt updated. ");
403                 System.out.println(e.getMessage());
404             }
405 
406             // This has to happen at this point as a forced minimum.
407             isLocalDatabaseUpToDate = true;
408         }
409     }
410 
411     public static void main(String[] args) {
412 
413         // This is so that the fonts are rendered correctly in Swing gui.
414         System.setProperty("awt.useSystemAAFontSettings", "on");
415         System.setProperty("swing.aatext", "true");
416 
417         // This is to call the thread, so we can check the databases.
418         Main t1 = new Main();
419         t1.start();
420 
421         // As the thread starts, we start the game. Usually it has to read from
    the
422         // backup file if the database isnt updated yet. After which it would
    start
423         // reading from there. As downloading the images and putting them in the
424         // database takes time and we cant wait that long, that job is
    multithreaded.
```

```
425         // The use of the backup database is :
426         // 1. It has some basic images that are shipped with the jar file so in
    case
427         // someone doesnt have internet, atleast they have something.
428         // 2. It is the fallback in case something goes wrong while doing or
    reading
429         // something from one of the files.
430         // 3. It serves as the Primary database when we are updating the local
    database,
431         // and we still need to show stuff to the user so they can play the game.
    This
432         // is the most important one.
433         loginFrame = new LoginFrame();
434     }
435 }
```

Listing 4: Main Java file

```
1
2 /*
3  * This is the loginFrame file, which is one of the first classes that comes into
     picture, pun intended.
4  * It does everything it can to make it look like the login screen of a modern
     website like google. Even goes as far as to use the same fonts.
5  * It checks the username and the password entered by the user, and matches it
     with the csv file it has. And reports the situation on screen.
6  * Once the user is justified and logged in, or has created a new account, it adds
      them, assigns some basic variables, and then Calls the Main Menu class by
     calling the changeFrame function in the Main class.
7  */
8
9 package org.howmuch;
10
11 import javax.swing.*;
12 import java.awt.*;
13 import java.awt.event.ActionEvent;
14 import java.awt.event.ActionListener;
15 import java.util.Arrays;
16
17 // Basically importing every static thing from Main coz its used so often
18 // not a very good practice.
19 import static org.howmuch.Main.*;
20
21 public class LoginFrame extends JFrame implements Runnable {
22
23     public static boolean running = true, userExists = false, incorrectPassword =
    false, newUser = false;
24     JLabel username_lbl, password_lbl, background_lbl, status_lbl,
    status_emoji_lbl;
25     JButton login_btn, guest_btn, newAccount_btn, exit_btn, resize_btn,
    minimize_btn;
26     JTextField username_txt_fld;
27     JPasswordField password_txt_fld;
28     Thread loginThread;
29
30     /*
31      * This is the standard implmentation of a constructor in this game. There are
32      * some basic attributes of the Frame calsses that it extends from
33      * And then sets them. It then creates the things you are supposed to create
```

```
in
34       * the GUI, and then adds them to the frame. This could be done in a panel,
    yes,
35       * but then you couldnt use some things like the CompoenentListener class that
36       * only listens to the Frame mainly, and that helps in calling certain
    functions
37       * when you resize the screen.
38       */
39      LoginFrame() {
40          this.setTitle("How Much? ");
41          this.setResizable(false);
42          this.setUndecorated(true);
43          this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
44
45          createFonts();
46          createButtons();
47          createLabels();
48          createTextFields();
49
50          this.add(status_lbl);
51          this.add(status_emoji_lbl);
52          this.add(login_btn);
53          this.add(username_lbl);
54          this.add(password_lbl);
55          this.add(guest_btn);
56          this.add(newAccount_btn);
57          this.add(exit_btn);
58          this.add(minimize_btn);
59          this.add(username_txt_fld);
60          this.add(password_txt_fld);
61          this.add(background_lbl);
62
63          // Thread to check the password entered by the user every 2 seconds is
    invoked
64          // by this thread's start method.
65          startThread();
66
67          this.pack();
68          this.setVisible(true);
69          this.setLocationRelativeTo(null); // put in the center.
70      }
71
72      /*
73       * Standard function to create buttons and assign their attributes. As some
74       * lines are dupiliated, they certainly can be extracted as separete methods
75       * themselves.
76       */
77      public void createButtons() {
78          login_btn = new JButton();
79          login_btn.setText("Login");
80          login_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
81          login_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
82          login_btn.setBounds(775, 532, 200, 40);
83          login_btn.setFont(textFont.deriveFont(44f));
84          login_btn.setFocusPainted(false);
85          login_btn.setContentAreaFilled(false);
86          login_btn.setOpaque(true);
87          login_btn.setBorder(null);
88
```

```
89          login_btn.setBackground(Colors.bgColor);
90
91          // To control what happens when the mouse interacts with this button.
92          login_btn.addChangeListener(evt -> {
93              if (login_btn.getModel().isPressed()) {
94                  login_btn.setForeground(Colors.primaryColor);
95              } else if (login_btn.getModel().isRollover()) {
96                  login_btn.setForeground(Colors.secondaryColor);
97              } else {
98                  login_btn.setForeground(Colors.primaryColor);
99              }
100         });
101
102         // Stuff to do when it is pressed. Applicable to all buttons here.
103         login_btn.addActionListener(e -> {
104             if (DataBaseManager.doesUsernameExist(username_txt_fld.getText())) {
105
106                 // Now the user is trying to login, so we check if the password
    matches
107                 // if it does, then we assign some basic variables, and close this
     screen, open
108                 // the menu screen.
109                 if (DataBaseManager.doesPasswordMatch(username_txt_fld.getText(),
110                         String.valueOf(password_txt_fld.getPassword()))) {
111                     DataBaseManager.currentUsername = username_txt_fld.getText();
112                     DataBaseManager.currentPassword = String.valueOf(
    password_txt_fld.getPassword());
113                     this.setVisible(false);
114                     this.dispose();
115                     running = false;
116                     grantAccess = true;
117                     Main.changeFrame(1);
118                 } else {
119                     grantAccess = false;
120                     incorrectPassword = true;
121                 }
122             } else {
123                 // If the user has entered a username and a password, and he
    doesnt exist, then
124                 // clearly is a new user.
125                 newUser = true;
126             }
127
128         });
129         login_btn.setEnabled(false);
130
131         guest_btn = new JButton();
132         guest_btn.setText("Guest");
133         guest_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
134         guest_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
135         guest_btn.setBounds(940, 532, 200, 40);
136         guest_btn.setFont(textFont.deriveFont(44f));
137         guest_btn.setFocusPainted(false);
138         guest_btn.setContentAreaFilled(false);
139         guest_btn.setOpaque(true);
140         guest_btn.setBorder(null);
141         guest_btn.setBackground(Colors.bgColor);
142         guest_btn.addChangeListener(evt -> {
143             if (guest_btn.getModel().isPressed()) {
```

```
144                 guest_btn.setForeground(Colors.primaryColor);
145             } else if (guest_btn.getModel().isRollover()) {
146                 guest_btn.setForeground(Colors.secondaryColor);
147             } else {
148                 guest_btn.setForeground(Colors.primaryColor);
149             }
150         });
151         guest_btn.addActionListener(e -> {
152             Main.isGuest = true;
153             DataBaseManager.currentPassword = "guest";
154             DataBaseManager.currentUsername = "guest";
155             this.setVisible(false);
156             this.dispose();
157             running = false;
158             grantAccess = true;
159             Main.changeFrame(1);
160         });
161 
162         newAccount_btn = new JButton();
163         newAccount_btn.setText("Create New Account");
164         newAccount_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
165         newAccount_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
166         newAccount_btn.setBounds(615, 590, 800, 40);
167         newAccount_btn.setFont(textFont.deriveFont(44f));
168         newAccount_btn.setFocusPainted(false);
169         newAccount_btn.setContentAreaFilled(false);
170         newAccount_btn.setOpaque(true);
171         newAccount_btn.setBorder(null);
172         newAccount_btn.setBackground(Colors.bgColor);
173         newAccount_btn.addChangeListener(evt -> {
174             if (newAccount_btn.getModel().isPressed()) {
175                 newAccount_btn.setForeground(Colors.primaryColor);
176             } else if (newAccount_btn.getModel().isRollover()) {
177                 newAccount_btn.setForeground(Colors.secondaryColor);
178             } else {
179                 newAccount_btn.setForeground(Colors.primaryColor);
180             }
181         });
182 
183         newAccount_btn.addActionListener(e -> {
184             Main.isGuest = false;
185             if (!DataBaseManager.doesUsernameExist(username_txt_fld.getText())) {
186                 DataBaseManager.currentUsername = username_txt_fld.getText();
187                 DataBaseManager.currentPassword = String.valueOf(password_txt_fld.
    getPassword());
188                 DataBaseManager.currentScore = 0;
189                 DataBaseManager.addNewUser();
190                 running = false;
191                 this.setVisible(false);
192                 this.dispose();
193                 grantAccess = true;
194                 Main.changeFrame(1);
195             } else {
196                 System.out.println("User Already Exists");
197                 userExists = true;
198             }
199         });
200         newAccount_btn.setEnabled(false);
201 
```

```
202         ImageIcon exit = new ImageIcon("/run/media/krishnaraj/Programs/Java/How
    Much/src/main/resources/icons/circle_delete.png");
203         Image exit_image = exit.getImage().getScaledInstance(25, 25, Image.
    SCALE_SMOOTH);
204         ImageIcon minimize = new ImageIcon("/run/media/krishnaraj/Programs/Java/
    How Much/src/main/resources/icons/circle_minus.png");
205         Image minimize_image = minimize.getImage().getScaledInstance(25, 25, Image
    .SCALE_SMOOTH);
206         ImageIcon resize = new ImageIcon("/run/media/krishnaraj/Programs/Java/How
    Much/src/main/resources/icons/screen_expand_3.png");
207         Image resize_image = resize.getImage().getScaledInstance(25, 25, Image.
    SCALE_SMOOTH);
208
209         exit_btn = new JButton();
210         // exit_btn.setText("-");
211         exit_btn.setIcon(new ImageIcon(exit_image));
212         exit_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
213         exit_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
214         exit_btn.setBounds(1230, 15, 25, 25);
215         exit_btn.setFont(textFont.deriveFont(44f));
216         exit_btn.setFocusPainted(false);
217         exit_btn.setContentAreaFilled(false);
218         exit_btn.setOpaque(true);
219         exit_btn.setBorder(null);
220         exit_btn.setBackground(Colors.bgColor);
221         exit_btn.addChangeListener(evt -> {
222             if (exit_btn.getModel().isPressed()) {
223                 exit_btn.setForeground(Colors.primaryColor);
224                 this.setVisible(false);
225                 this.dispose();
226                 running = false;
227                 Main.changeFrame(0);
228             } else if (exit_btn.getModel().isRollover()) {
229                 exit_btn.setForeground(Colors.secondaryColor);
230             } else {
231                 exit_btn.setForeground(Colors.primaryColor);
232             }
233         });
234
235         resize_btn = new JButton();
236         resize_btn.setIcon(new ImageIcon(resize_image));
237         resize_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
238         resize_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
239         resize_btn.setBounds(1195, 15, 25, 25);
240         resize_btn.setFont(textFont.deriveFont(44f));
241         resize_btn.setFocusPainted(false);
242         resize_btn.setContentAreaFilled(false);
243         resize_btn.setOpaque(true);
244         resize_btn.setBorder(null);
245         resize_btn.setBackground(Colors.bgColor);
246         resize_btn.addChangeListener(evt -> {
247             if (exit_btn.getModel().isPressed()) {
248                 this.setExtendedState(JFrame.MAXIMIZED_BOTH);
249                 exit_btn.setForeground(Colors.primaryColor);
250             } else if (exit_btn.getModel().isRollover()) {
251                 exit_btn.setForeground(Colors.secondaryColor);
252             } else {
253                 exit_btn.setForeground(Colors.primaryColor);
254             }
```

```
255         });
256 
257         minimize_btn = new JButton();
258         // minimize_btn.setText("-");
259         minimize_btn.setIcon(new ImageIcon(minimize_image));
260         minimize_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
261         minimize_btn.setAlignmentX(Box.CENTER_ALIGNMENT);
262         minimize_btn.setBounds(1195, 15, 25, 25);
263         minimize_btn.setFont(textFont.deriveFont(44f));
264         minimize_btn.setFocusPainted(false);
265         minimize_btn.setContentAreaFilled(false);
266         minimize_btn.setOpaque(true);
267         minimize_btn.setBorder(null);
268         minimize_btn.setBackground(Colors.bgColor);
269         minimize_btn.addChangeListener(evt -> {
270             if (minimize_btn.getModel().isPressed()) {
271                 this.setState(JFrame.ICONIFIED);
272                 minimize_btn.setForeground(Colors.primaryColor);
273             } else if (minimize_btn.getModel().isRollover()) {
274                 minimize_btn.setForeground(Colors.secondaryColor);
275             } else {
276                 minimize_btn.setForeground(Colors.primaryColor);
277             }
278         });
279 
280     }
281 
282     /*
283      * Standard function to create labels used in this frame.
284      */
285     public void createLabels() {
286         ImageIcon icon = new ImageIcon("src/main/resources/images/Login_bg.png");
287         Image bg_image = icon.getImage().getScaledInstance(1280, 720, Image.
    SCALE_SMOOTH);
288 
289         background_lbl = new JLabel();
290         background_lbl.setIcon(new ImageIcon(bg_image));
291 
292         username_lbl = new JLabel();
293         username_lbl.setText("Enter Username");
294         username_lbl.setFont(textFont.deriveFont(44f));
295         username_lbl.setAlignmentX(Box.CENTER_ALIGNMENT);
296         username_lbl.setBounds(822, 244, 800, 80);
297         username_lbl.setForeground(Colors.primaryColor);
298 
299         password_lbl = new JLabel();
300         password_lbl.setText("Password");
301         password_lbl.setFont(textFont.deriveFont(44f));
302         password_lbl.setAlignmentX(Box.CENTER_ALIGNMENT);
303         password_lbl.setBounds(822, 340, 800, 150);
304         password_lbl.setForeground(Colors.primaryColor);
305 
306         status_lbl = new JLabel();
307         status_lbl.setText("");
308         status_lbl.setFont(password_font.deriveFont(30f).deriveFont(Font.ITALIC));
309         status_lbl.setAlignmentX(Box.CENTER_ALIGNMENT);
310         status_lbl.setBounds(30, 580, 800, 40);
311         status_lbl.setForeground(Colors.bgColor);
312 
```

```
313         status_emoji_lbl = new JLabel();
314         status_emoji_lbl.setText("");
315         status_emoji_lbl.setFont(emoji_font.deriveFont(50f));
316         status_emoji_lbl.setAlignmentX(Box.CENTER_ALIGNMENT);
317         status_emoji_lbl.setBounds(25, 500, 80, 80);
318         status_emoji_lbl.setForeground(Colors.bgColor);
319     }
320 
321     /*
322      * Standard function to create Text Fields.
323      */
324     public void createTextFields() {
325         username_txt_fld = new JTextField("");
326         username_txt_fld.setFont(password_font);
327         username_txt_fld.setBounds(822, 319, 300, 50);
328         username_txt_fld.setBackground(Colors.bgColor);
329         username_txt_fld.setOpaque(true);
330         username_txt_fld.setBorder(null);
331         username_txt_fld.setForeground(Colors.accentColor);
332         username_txt_fld.addActionListener(new ActionListener() {
333             @Override
334             public void actionPerformed(ActionEvent e) {
335                 System.out.println(username_txt_fld.getText());
336             }
337         });
338 
339         password_txt_fld = new JPasswordField("");
340         password_txt_fld.setFont(password_font);
341         password_txt_fld.setBounds(822, 452, 300, 50);
342         password_txt_fld.setBackground(Colors.bgColor);
343         password_txt_fld.setOpaque(true);
344         password_txt_fld.setBorder(null);
345         password_txt_fld.setEchoChar('*');
346         password_txt_fld.setForeground(Colors.accentColor);
347         password_txt_fld.setAlignmentY(Box.CENTER_ALIGNMENT);
348     }
349 
350     /*
351      * This function is what checks the password, and so naturally has a lot of if
352      * statements.
353      * Most of them are self explanatory.
354      */
355     @Override
356     public void run() {
357         long lastTime = System.nanoTime();
358         double amountOfTicks = 5.00;
359         double ns = 1000000000 / amountOfTicks;
360         double delta = 0;
361 
362         while (running) {
363             // basic game loop logic to ensure 60 fps, dont think too much about
    it, it
364             // makes sense.
365             // you can reuse it for consistancy, or make a new one.
366 
367             long now = System.nanoTime();
368             delta += (now - lastTime) / ns;
369             lastTime = now;
370 
```

```
371             if (delta >= 1) {
372                 // System.out.println(username_txt_fld.getText());
373                 // System.out.println(password_txt_fld.getPassword());
374                 if (username_txt_fld.getText().length() == 0) {
375                     newAccount_btn.setEnabled(false);
376                     status_lbl.setText("Enter Username & Password");
377                     status_emoji_lbl.setText("\uD83E\uDEE3");
378                     // status_emoji_lbl.setText("");
379                 } else if (newUser) {
380                     status_lbl.setText("Welcome! Create New Account");
381                     status_emoji_lbl.setText("\uD83D\uDE4F");
382                     // status_emoji_lbl.setText("\uD83D\uDE1E");
383                     try {
384                         Thread.sleep(2000);
385                     } catch (InterruptedException e) {
386                         throw new RuntimeException(e);
387                     }
388                     newUser = false;
389                 } else if (incorrectPassword) {
390                     status_lbl.setText("Password doesn't Match!");
391                     status_emoji_lbl.setText("\uD83D\uDE16");
392                     // status_emoji_lbl.setText("\uD83D\uDE1E");
393                     try {
394                         Thread.sleep(2000);
395                     } catch (InterruptedException e) {
396                         throw new RuntimeException(e);
397                     }
398                     incorrectPassword = false;
399                 } else if (userExists) {
400                     status_lbl.setText("User Already Exists, Try to Login");
401                     status_emoji_lbl.setText("\uD83D\uDE15");
402                     try {
403                         Thread.sleep(2000);
404                     } catch (InterruptedException e) {
405                         throw new RuntimeException(e);
406                     }
407                     userExists = false;
408                 } else if (password_txt_fld.getPassword().length < 8) {
409                     newAccount_btn.setEnabled(false);
410                     status_lbl.setText("Nope, Password is too Short");
411                      status_emoji_lbl.setText("\uD83D\uDE0F");
412                     // status_emoji_lbl.setText("\uD83E\uDD0F");
413 //                    status_emoji_lbl.setText("\uD83D\uDE15");
414                 } else if (Arrays.equals(password_txt_fld.getPassword(), "abcdefgh
    ".toCharArray())) {
415                     newAccount_btn.setEnabled(false);
416                     status_lbl.setText("Anyone can guess that bruh");
417                     status_emoji_lbl.setText("\uD83D\uDC80");
418                 } else if (Arrays.equals(password_txt_fld.getPassword(), "12345678
    ".toCharArray())) {
419                     newAccount_btn.setEnabled(false);
420                     status_lbl.setText("12345678? Really?");
421                     status_emoji_lbl.setText("\uD83E\uDEE0");
422                 } else if (Arrays.equals(password_txt_fld.getPassword(), "asdfghjk
    ".toCharArray())) {
423                     newAccount_btn.setEnabled(false);
424                     status_lbl.setText("Be Lazy, but not thaaat lazy");
425                     // status_emoji_lbl.setText("\uD83D\uDE42");
426                     status_emoji_lbl.setText("\uD83D\uDC80");
```

```
427                 } else if (Arrays.equals(password_txt_fld.getPassword(), "asdfasdf
    ".toCharArray())) {
428                     newAccount_btn.setEnabled(false);
429                     status_lbl.setText("Even Krishnaraj isnt this lazy");
430                     // status_emoji_lbl.setText("\uD83E\uDD21");
431                     status_emoji_lbl.setText("\uD83D\uDC80");
432                 } else if (password_txt_fld.getPassword().length > 30) {
433                     newAccount_btn.setEnabled(false);
434                     status_lbl.setText("Woah, Its too big");
435                     status_emoji_lbl.setText("\uD83D\uDE0F");
436                 } else {
437                     status_lbl.setText("All Good!");
438                     status_emoji_lbl.setText("\uD83D\uDC4C");
439                     // status_emoji_lbl.setText("\uD83D\uDE0E" );
440                     // status_emoji_lbl.setText("" );
441                     newAccount_btn.setEnabled(true);
442                     login_btn.setEnabled(true);
443                 }
444                 delta--;
445             }
446 
447         }
448     }
449 
450     public void startThread() {
451 
452         // Creating the Game Thread
453         loginThread = new Thread(this);
454         loginThread.start();
455 
456     }
457 }
```

Listing 5: Main Java file

```
1  package org.howmuch;
2 
3  import javax.swing.*;
4  import javax.swing.event.ChangeEvent;
5  import javax.swing.event.ChangeListener;
6  import java.awt.*;
7  import java.awt.event.*;
8  import java.util.ArrayList;
9  import java.util.Arrays;
10 import java.util.Random;
11 import java.util.TimerTask;
12 import java.util.Timer;
13 
14 import static java.lang.Math.round;
15 import static org.howmuch.Main.*;
16 
17 public class GameFrame extends JFrame {
18     static Timer timer;
19     public static int time_left = 9;
20     public static boolean gameWon = false;
21     static boolean[] whichOptionCorrect;
22     BackgroundPanel backgroundPanel;
23     BackgroundPanel productImagePanel;
24     static JButton option_1_btn;
```

```
    static JButton option_2_btn;
    static JButton option_3_btn;
    static JButton option_4_btn;
    JPanel options_panel;
    static JLabel time_lbl;
    JTextArea productName_txtArea;
    public static int randomIndex = 0;
    static String[] currentData;
    static int correctPrice;
    static JLabel confetti;

    GameFrame() {
        randomIndex = 0;
        time_left = 9;
        backgroundPanel = new BackgroundPanel();
        gameWon = false;
        this.setTitle("How Much?");
        if (maximized) {
            this.setExtendedState(MAXIMIZED_BOTH);
        } else {
            this.setPreferredSize(new Dimension(Main.WIDTH, Main.HEIGHT));
        }
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setResizable(true);
        this.setUndecorated(true);
        this.setMinimumSize(new Dimension(1280, 720));

        createFonts();
        createBasicButtonPanel();
        createButtons();
        createPanels();
        createLabels();


        // Main stuff
        findRandomIndex();
        assignCurrentData();

        // Important Updates
        reassignColors();
        reassignBounds();

        startTimer();


        this.addComponentListener(new ComponentAdapter() {
            @Override
            public void componentResized(ComponentEvent e) {
                reassignBounds();
                repaint();
            }
        });

        this.add(confetti);
        this.add(productName_txtArea);
        this.add(productImagePanel);
        this.add(time_lbl);
        this.add(options_panel);
        this.add(basicButtons_pnl);
```

```
84          this.add(backgroundPanel);
85          this.pack();
86          this.setLocationRelativeTo(null);
87          this.setVisible(true);
88      }
89
90      public static void startTimer() {
91          timer = new Timer();
92          timer.schedule(new TimerTask() {
93              @Override
94              public void run() {
95                  System.out.println(time_left);
96                  if (time_left == 0) {
97                      timer.cancel();
98                      timer.purge();
99                      grantAccess = true;
100                     Main.changeFrame(7);
101                 }
102                 time_left--;
103                 changeTimeOnTimer();
104             }
105         }, 1000, 1000);
106     }
107
108     public static void changeTimeOnTimer() {
109         time_lbl.setText(String.valueOf(time_left));
110     }
111
112     private void assignCurrentData() {
113         currentData = new String[]{"", "", ""};
114         System.out.println("Accessing or atleast trying to access data here");
115         try {
116             if (!usingMongo) {
117                 System.out.println("We are accessing data from the local database
    as mongo isnt working");
118                 currentData = DataBaseManager.readFromLocalDatabase(currentTopic,
    randomIndex);
119             } else {
120                 currentData = MongoManager.fetchDataFromMongo(currentTopic,
    randomIndex);
121                 System.out.println("Reading data from mongo sucessful");
122             }
123         } catch (Exception e) {
124             System.out.println("We got some Issues reading the file from Mongodb")
    ;
125             currentData = DataBaseManager.readFromLocalDatabase(currentTopic,
    randomIndex);
126         }
127     }
128
129     private void findRandomIndex() {
130         int max = DataBaseManager.findLength(currentTopic);
131         Random random = new Random();
132         // Generates random integers 0 to 49
133         randomIndex = random.nextInt(max);
134     }
135
136     private void loadGameDataOnScreen() {
137         System.out.println(Arrays.toString(currentData));
```

```
138         Image productImage = new ImageIcon(currentData[2]).getImage();
139         int maxWidth = (int) (0.4 * this.getWidth());
140         int maxHeight = (int) (0.4 * this.getWidth());
141 
142         int[] imageSize = calculateImageSize(maxWidth, maxHeight, productImage.
    getWidth(productImagePanel), productImage.getHeight(productImagePanel));
143         System.out.println(Arrays.toString(imageSize));
144         productImagePanel.setBounds((int) (0.07 * this.getWidth()) + maxWidth / 2
    - imageSize[0] / 2, (int) (0.07 * this.getHeight()) + maxHeight / 2 - imageSize
    [1] / 2, imageSize[0], imageSize[1]);
145         productImagePanel.setBackground(currentData[2]);
146 
147         productName_txtArea.setBounds((int) (0.065 * this.getWidth()), (int) (0.83
     * this.getHeight()), (int) (0.5 * this.getWidth()), (int) (0.2 * this.
    getHeight()));
148         productName_txtArea.setText(currentData[0]);
149 
150         // setting price
151         setPrices();
152     }
153 
154     public static void setPrices() {
155         Random random = new Random();
156         int[] wrongPrices = new int[]{0, 0, 0};
157         correctPrice = Math.round(Integer.parseInt(currentData[1])) + 2;
158         System.out.println("Correct price is: ");
159         System.out.println(correctPrice);
160 
161         double randomMultiplier = 0.3 + random.nextDouble(2.7);
162         System.out.println(randomMultiplier);
163         wrongPrices[0] = (int) (correctPrice * randomMultiplier);
164 
165         randomMultiplier = 0.3 + random.nextDouble(2.7);
166         System.out.println(randomMultiplier);
167         wrongPrices[1] = (int) (correctPrice * randomMultiplier);
168 
169         randomMultiplier = 0.3 + random.nextDouble(2.7);
170         System.out.println(randomMultiplier);
171         wrongPrices[2] = (int) (correctPrice * randomMultiplier);
172 
173         ArrayList<Integer> list = new ArrayList<Integer>(4);
174         list.add(correctPrice);
175         for (int i = 0; i < 3; i++) {
176             list.add(wrongPrices[i]);
177         }
178         System.out.println(list);
179 
180 //         option_1_btn.setText("R" + String.format("%,.0f", (double)
181 //                 round((double) list.remove(random.nextInt(list.size())) / 100) *
     100
182 //         ));
183 //         option_2_btn.setText("R" + String.format("%,.0f", (double)
184 //                 round((double) list.remove(random.nextInt(list.size())) / 100) *
     100
185 //         ));
186 //         option_3_btn.setText("R" + String.format("%,.0f", (double)
187 //                 round((double) list.remove(random.nextInt(list.size())) / 100) *
     100
188 //         ));
```

```
189 //         option_4_btn.setText("R" + String.format("%,.0f", (double)
190 //                 round((double) list.remove(random.nextInt(list.size())) / 100) *
        100
191 //         ));
192         whichOptionCorrect = new boolean[]{false, false, false, false};
193         int optionValue = 0;
194 
195         optionValue = list.remove(random.nextInt(list.size()));
196         whichOptionCorrect[0] = optionValue == correctPrice;
197         option_1_btn.setText("R" + String.format("%,.0f", (double) optionValue));
198 
199         optionValue = list.remove(random.nextInt(list.size()));
200         whichOptionCorrect[1] = optionValue == correctPrice;
201         option_2_btn.setText("R" + String.format("%,.0f", (double) optionValue));
202 
203         optionValue = list.remove(random.nextInt(list.size()));
204         whichOptionCorrect[2] = optionValue == correctPrice;
205         option_3_btn.setText("R" + String.format("%,.0f", (double) optionValue));
206 
207         optionValue = list.remove(random.nextInt(list.size()));
208         whichOptionCorrect[3] = optionValue == correctPrice;
209         option_4_btn.setText("R" + String.format("%,.0f", (double) optionValue));
210 
211     }
212 
213     private int[] calculateImageSize(int maxWidth, int maxHeight, double width,
        double height) {
214         int wt = 600, ht = 550;
215         double aspectRatio = width / height;
216         System.out.println(aspectRatio);
217         if (aspectRatio > 1) {
218             // landscape
219             wt = maxWidth;
220             ht = (int) (wt / aspectRatio);
221         } else {
222             // portrait image
223             ht = maxHeight;
224             wt = (int) (ht * aspectRatio);
225         }
226 
227         return new int[]{wt, ht};
228     }
229 
230     private void createLabels() {
231         time_lbl = new JLabel();
232         time_lbl.setAlignmentY(Box.CENTER_ALIGNMENT);
233         time_lbl.setAlignmentX(Box.CENTER_ALIGNMENT);
234         time_lbl.setOpaque(true);
235         time_lbl.setBorder(null);
236         time_lbl.setText(String.valueOf(time_left));
237 
238         productName_txtArea = new JTextArea();
239         productName_txtArea.setAlignmentY(Box.CENTER_ALIGNMENT);
240         productName_txtArea.setAlignmentX(Box.LEFT_ALIGNMENT);
241         productName_txtArea.setOpaque(true);
242         productName_txtArea.setBorder(null);
243         productName_txtArea.setLineWrap(true);
244 
245         ImageIcon imageIcon = new ImageIcon("src/main/resources/images/confetti.
```

```
gif");
246         confetti = new JLabel(imageIcon);
247         confetti.setVisible(false);
248     }
249 
250     private void reassignBounds() {
251         Dimension screenSize = this.getSize();
252 
253         // The Entire basic button panel for closing minimizing and stuff
254         basicButtons_pnl.setBounds(this.getWidth() - (exit_btn.getWidth() * 3) -
    40, 10, exit_btn.getWidth() * 3 + 35, exit_btn.getHeight());
255 
256         // Options panel
257         options_panel.setBounds((int) (0.70 * screenSize.getWidth()), (int) (0.58
    * screenSize.getHeight()), (int) (0.60 * screenSize.getWidth()), 700);
258 
259         // Buttons in the Options Panel
260         option_1_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
     80));
261         option_1_btn.setFont(options_font.deriveFont((float) (0.05 * getHeight()))
    );
262 
263         option_2_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
     80));
264         option_2_btn.setFont(options_font.deriveFont((float) (0.05 * getHeight()))
    );
265 
266         option_3_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
     70));
267         option_3_btn.setFont(options_font.deriveFont((float) (0.05 * getHeight()))
    );
268 
269         option_4_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
     70));
270         option_4_btn.setFont(options_font.deriveFont((float) (0.05 * getHeight()))
    );
271 
272         productName_txtArea.setFont(password_font.deriveFont((float) (0.04 *
    getHeight())));
273 
274         // The Score label
275         time_lbl.setBounds((int) (0.890 * screenSize.getWidth()), (int) (0.14 *
    screenSize.getHeight()), (int) (0.05 * screenSize.getWidth()), (int) (0.11 *
    screenSize.getHeight()));
276         time_lbl.setFont(buttonFont.deriveFont((float) (0.09 * getHeight())));
277 
278         confetti.setBounds((int) (-0.3 * screenSize.getWidth()), (int) (-0.27 *
    screenSize.getHeight()), this.getWidth(), this.getHeight());
279         loadGameDataOnScreen();
280     }
281 
282     private void reassignColors() {
283 
284         if (Colors.DarkMode) {
285             backgroundPanel.setBackground("src/main/resources/images/gamescreen.
    png");
286         } else {
287             backgroundPanel.setBackground("src/main/resources/images/gamescreen.
    png");
```

```
288         }
289         Colors.reassignColors();
290         basicButtons_pnl.setBackground(Colors.light_primaryColor);
291         exit_btn.setBackground(Colors.light_primaryColor);
292         resize_btn.setBackground(Colors.light_primaryColor);
293         minimize_btn.setBackground(Colors.light_primaryColor);
294         option_1_btn.setBackground(Colors.light_primaryColor);
295         option_1_btn.setForeground(Colors.light_bgColor);
296         option_2_btn.setBackground(Colors.light_primaryColor);
297         option_2_btn.setForeground(Colors.light_bgColor);
298         option_4_btn.setBackground(Colors.light_primaryColor);
299         option_4_btn.setForeground(Colors.light_bgColor);
300         option_3_btn.setBackground(Colors.light_primaryColor);
301         option_3_btn.setForeground(Colors.light_bgColor);
302         productName_txtArea.setBackground(Color.WHITE);
303         productName_txtArea.setForeground(Colors.light_primaryColor);
304         if (Colors.DarkMode) {
305             time_lbl.setBackground(Colors.light_secondaryColor);
306         } else {
307             time_lbl.setBackground(Colors.light_secondaryColor);
308         }
309         time_lbl.setForeground(Colors.light_primaryColor);
310     }
311 
312     private void createPanels() {
313         options_panel = new JPanel();
314         BoxLayout bl = new BoxLayout(options_panel, BoxLayout.Y_AXIS);
315         options_panel.setLayout(bl);
316         options_panel.add(option_1_btn);
317         options_panel.add(Box.createRigidArea(new Dimension(0, 25)));
318         options_panel.add(option_2_btn);
319         options_panel.add(Box.createRigidArea(new Dimension(0, 25)));
320         options_panel.add(option_3_btn);
321         options_panel.add(Box.createRigidArea(new Dimension(0, 25)));
322         options_panel.add(option_4_btn);
323         options_panel.setBackground(new Color(0, 0, 0, 0));
324 
325         productImagePanel = new BackgroundPanel();
326     }
327 
328     private void createButtons() {
329 
330         // Removing Change and Action Listeners.
331         removeAllChangeAndActionListenersFromBasicButtons();
332         exit_btn.addChangeListener(evt -> {
333             if (exit_btn.getModel().isPressed()) {
334                 timer.cancel();
335                 timer.purge();
336                 exit_btn.setForeground(Colors.primaryColor);
337                 this.setVisible(false);
338                 this.dispose();
339                 Main.changeFrame(0);
340             } else if (exit_btn.getModel().isRollover()) {
341                 exit_btn.setForeground(Colors.secondaryColor);
342             } else {
343                 exit_btn.setForeground(Colors.primaryColor);
344             }
345         });
346         resize_btn.addActionListener(e -> {
```

```
347             if (!Main.maximized) {
348                 this.setExtendedState(MAXIMIZED_BOTH);
349                 resize_btn.setIcon(new ImageIcon(resizeDown_image));
350             } else {
351                 this.setExtendedState(JFrame.NORMAL);
352                 this.setLocationRelativeTo(null);
353                 Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
354                 int x = (int) ((dimension.getWidth() - Main.WIDTH) / 2);
355                 int y = (int) ((dimension.getHeight() - Main.HEIGHT) / 2);
356                 this.setBounds(x, y, Main.WIDTH, Main.HEIGHT);
357                 resize_btn.setIcon(new ImageIcon(resizeUp_image));
358             }
359             Main.maximized = !Main.maximized;
360         });
361 
362         minimize_btn.addChangeListener(evt -> {
363             if (minimize_btn.getModel().isPressed()) {
364                 this.setState(JFrame.ICONIFIED);
365                 minimize_btn.setForeground(Colors.primaryColor);
366             } else if (minimize_btn.getModel().isRollover()) {
367                 minimize_btn.setForeground(Colors.secondaryColor);
368             } else {
369                 minimize_btn.setForeground(Colors.primaryColor);
370             }
371         });
372 
373         option_1_btn = new JButton();
374         assignButtonProperties(option_1_btn);
375         option_2_btn = new JButton();
376         assignButtonProperties(option_2_btn);
377         option_3_btn = new JButton();
378         assignButtonProperties(option_3_btn);
379         option_4_btn = new JButton();
380         assignButtonProperties(option_4_btn);
381     }
382 
383     static void removeAllChangeAndActionListenersFromBasicButtons() {
384         for (ActionListener listener : exit_btn.getActionListeners()) {
385             exit_btn.removeActionListener(listener);
386         }
387         for (ChangeListener listener : exit_btn.getChangeListeners()) {
388             exit_btn.removeChangeListener(listener);
389         }
390 
391         for (ActionListener listener : resize_btn.getActionListeners()) {
392             resize_btn.removeActionListener(listener);
393         }
394         for (ChangeListener listener : resize_btn.getChangeListeners()) {
395             resize_btn.removeChangeListener(listener);
396         }
397 
398         for (ActionListener listener : minimize_btn.getActionListeners()) {
399             minimize_btn.removeActionListener(listener);
400         }
401         for (ChangeListener listener : minimize_btn.getChangeListeners()) {
402             minimize_btn.removeChangeListener(listener);
403         }
404     }
405 
```

```
406     static void removeChangeAndActionListenersFromOption_btns() {
407         for (ChangeListener changeListener : option_1_btn.getChangeListeners()) {
408             option_1_btn.removeChangeListener(changeListener);
409         }
410         for (ChangeListener changeListener : option_2_btn.getChangeListeners()) {
411             option_2_btn.removeChangeListener(changeListener);
412         }
413         for (ChangeListener changeListener : option_3_btn.getChangeListeners()) {
414             option_3_btn.removeChangeListener(changeListener);
415         }
416         for (ChangeListener changeListener : option_4_btn.getChangeListeners()) {
417             option_4_btn.removeChangeListener(changeListener);
418         }
419 
420         for (ActionListener ActionListener : option_1_btn.getActionListeners()) {
421             option_1_btn.removeActionListener(ActionListener);
422         }
423         for (ActionListener ActionListener : option_2_btn.getActionListeners()) {
424             option_2_btn.removeActionListener(ActionListener);
425         }
426         for (ActionListener ActionListener : option_3_btn.getActionListeners()) {
427             option_3_btn.removeActionListener(ActionListener);
428         }
429         for (ActionListener ActionListener : option_4_btn.getActionListeners()) {
430             option_4_btn.removeActionListener(ActionListener);
431         }
432     }
433 
434 
435     private void assignButtonProperties(JButton optionButton) {
436         optionButton.setText("");
437         optionButton.setAlignmentY(Box.CENTER_ALIGNMENT);
438         optionButton.setAlignmentX(Box.LEFT_ALIGNMENT);
439         optionButton.setFocusPainted(false);
440         optionButton.setBounds(0, 0, 500, 500);
441         optionButton.setContentAreaFilled(false);
442         optionButton.setOpaque(true);
443         optionButton.setBorder(null);
444         optionButton.addChangeListener(evt -> {
445             if (optionButton.getModel().isPressed()) {
446                 optionButton.setForeground(Colors.accentColor);
447             } else if (optionButton.getModel().isRollover()) {
448                 optionButton.setForeground(Colors.accentColor);
449             } else {
450                 optionButton.setForeground(Colors.light_bgColor);
451             }
452         });
453         optionButton.addActionListener(e -> {
454             int this_btn_price = Integer.parseInt(optionButton.getText().replace("
    R", "").replace(",", ""));
455             if (correctPrice == this_btn_price) {
456                 runWinningClosingErrands();
457             } else {
458                 runLosingClosingErrands(this_btn_price);
459             }
460         });
461     }
462 
463     private void runLosingClosingErrands(int this_btn_price) {
```

```
464         removeChangeAndActionListenersFromOption_btns();
465         if (whichOptionCorrect[0]) {
466             option_1_btn.setForeground(new Color(56, 159, 82));
467 //            option_1_btn.setFont(options_font.deriveFont((float) (0.05 *
    getHeight())).deriveFont(Font.BOLD));
468             option_2_btn.setForeground(new Color(227, 83, 83));
469             option_3_btn.setForeground(new Color(227, 83, 83));
470             option_4_btn.setForeground(new Color(227, 83, 83));
471         } else if (whichOptionCorrect[1]) {
472             option_2_btn.setForeground(new Color(56, 159, 82));
473 //            option_2_btn.setFont(options_font.deriveFont((float) (0.05 *
    getHeight())).deriveFont(Font.BOLD));
474             option_1_btn.setForeground(new Color(227, 83, 83));
475             option_3_btn.setForeground(new Color(227, 83, 83));
476             option_4_btn.setForeground(new Color(227, 83, 83));
477         } else if (whichOptionCorrect[2]) {
478             option_3_btn.setForeground(new Color(56, 159, 82));
479 //            option_3_btn.setFont(options_font.deriveFont((float) (0.05 *
    getHeight())).deriveFont(Font.BOLD));
480             option_2_btn.setForeground(new Color(227, 83, 83));
481             option_1_btn.setForeground(new Color(227, 83, 83));
482             option_4_btn.setForeground(new Color(227, 83, 83));
483         } else if (whichOptionCorrect[3]) {
484             option_4_btn.setForeground(new Color(56, 159, 82));
485 //            option_4_btn.setFont(options_font.deriveFont((float) (0.05 *
    getHeight())).deriveFont(Font.BOLD));
486             option_2_btn.setForeground(new Color(227, 83, 83));
487             option_3_btn.setForeground(new Color(227, 83, 83));
488             option_1_btn.setForeground(new Color(227, 83, 83));
489         }
490         System.out.println("That was an incorrect Guess");
491         System.out.println(this_btn_price);
492         time_left = 3;
493         gameWon = false;
494         grantAccess = true;
495     }
496 
497     private void runWinningClosingErrands() {
498         removeChangeAndActionListenersFromOption_btns();
499         System.out.println("You guessed correctly");
500         DataBaseManager.currentScore += time_left;
501         confetti.setVisible(true);
502         time_left = 2;
503         grantAccess = true;
504         gameWon = true;
505     }
506 }
```

Listing 6: Main Java file

```
1 package org.howmuch;
2 
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.*;
6 
7 import static org.howmuch.Main.*;
8 
9 public class TopicsFrame extends JFrame {
```

```
10
11      BackgroundPanel backgroundPanel;
12      JButton option1_btn, option2_btn, option3_btn, option4_btn;
13      JButton backToMenu_btn;
14      JPanel options_panel;
15
16      TopicsFrame() {
17          backgroundPanel = new BackgroundPanel();
18
19          this.setTitle("How Much?");
20          if (maximized) {
21              this.setExtendedState(MAXIMIZED_BOTH);
22          } else {
23              this.setPreferredSize(new Dimension(Main.WIDTH, Main.HEIGHT));
24          }
25          this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
26          this.setResizable(true);
27          this.setUndecorated(true);
28          this.setMinimumSize(new Dimension(1280, 720));
29
30          createFonts();
31          createBasicButtonPanel();
32          createButtons();
33          createPanels();
34          reassignColors();
35          reassignBounds();
36
37          this.addComponentListener(new ComponentAdapter() {
38              @Override
39              public void componentResized(ComponentEvent e) {
40                  reassignBounds();
41                  repaint();
42              }
43          });
44
45          this.add(backToMenu_btn);
46          this.add(options_panel);
47          this.add(basicButtons_pnl);
48          this.add(backgroundPanel);
49          this.pack();
50          this.setLocationRelativeTo(null);
51          this.setVisible(true);
52      }
53
54      private void reassignBounds() {
55          Dimension screenSize = this.getSize();
56
57          // The back to menu mode label
58          backToMenu_btn.setBounds((int) (0.015 * screenSize.getWidth()), (int)
    (0.80 * screenSize.getHeight()),
59                  (int) (0.20 * screenSize.getWidth()), (int) (0.07 * screenSize.
    getHeight()));
60          backToMenu_btn.setFont(buttonFont.deriveFont((float) (0.05 * getHeight()))
    );
61
62          // The Entire basic button panel for closing minimizing and stuff
63          basicButtons_pnl.setBounds(this.getWidth() - (exit_btn.getWidth() * 3) -
    40, 10, exit_btn.getWidth() * 3 + 35,
64                  exit_btn.getHeight());
```

```
65
66          // Options panel
67          options_panel.setBounds((int) (0.60 * screenSize.getWidth()), (int) (0.34
    * screenSize.getHeight()),
68                  (int) (0.45 * screenSize.getWidth()), 700);
69
70          // Buttons in the Options Panel
71          option1_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
    80));
72          option1_btn.setFont(buttonFont.deriveFont((float) (0.07 * getHeight())));
73
74          option2_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
    80));
75          option2_btn.setFont(buttonFont.deriveFont((float) (0.07 * getHeight())));
76
77          option3_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
    70));
78          option3_btn.setFont(buttonFont.deriveFont((float) (0.07 * getHeight())));
79
80          option4_btn.setBounds(new Rectangle((int) (0.45 * screenSize.getWidth()),
    70));
81          option4_btn.setFont(buttonFont.deriveFont((float) (0.07 * getHeight())));
82      }
83
84      private void reassignColors() {
85
86          if (Colors.DarkMode) {
87              backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java/How
     Much/src/main/resources/images/choose topic dark.png");
88          } else {
89              backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java/How
     Much/src/main/resources/images/choose topic.png");
90          }
91          Colors.reassignColors();
92          basicButtons_pnl.setBackground(Colors.bgColor);
93          exit_btn.setBackground(Colors.bgColor);
94          resize_btn.setBackground(Colors.bgColor);
95          backToMenu_btn.setBackground(Colors.primaryColor);
96          backToMenu_btn.setForeground(Colors.bgColor);
97          minimize_btn.setBackground(Colors.bgColor);
98          option1_btn.setBackground(Colors.bgColor);
99          option1_btn.setForeground(Colors.primaryColor);
100         option2_btn.setBackground(Colors.bgColor);
101         option2_btn.setForeground(Colors.primaryColor);
102         option4_btn.setBackground(Colors.bgColor);
103         option4_btn.setForeground(Colors.primaryColor);
104         option3_btn.setBackground(Colors.bgColor);
105         option3_btn.setForeground(Colors.primaryColor);
106     }
107
108     private void createPanels() {
109         options_panel = new JPanel();
110         BoxLayout bl = new BoxLayout(options_panel, BoxLayout.Y_AXIS);
111         options_panel.setLayout(bl);
112         options_panel.add(option1_btn);
113         options_panel.add(Box.createRigidArea(new Dimension(0, 25)));
114         options_panel.add(option2_btn);
115         options_panel.add(Box.createRigidArea(new Dimension(0, 25)));
116         options_panel.add(option3_btn);
```

```
117         options_panel.add(Box.createRigidArea(new Dimension(0, 25)));
118         options_panel.add(option4_btn);
119         options_panel.setBackground(new Color(0, 0, 0, 0));
120 
121     }
122 
123     private void createButtons() {
124 
125         // Removing Change and Action Listeners.
126         GameFrame.removeAllChangeAndActionListenersFromBasicButtons();
127 
128         exit_btn.addChangeListener(evt -> {
129             if (exit_btn.getModel().isPressed()) {
130                 exit_btn.setForeground(Colors.primaryColor);
131                 Main.changeFrame(0);
132             } else if (exit_btn.getModel().isRollover()) {
133                 exit_btn.setForeground(Colors.secondaryColor);
134             } else {
135                 exit_btn.setForeground(Colors.primaryColor);
136             }
137         });
138         resize_btn.addActionListener(e -> {
139             if (!Main.maximized) {
140                 this.setExtendedState(MAXIMIZED_BOTH);
141                 resize_btn.setIcon(new ImageIcon(resizeDown_image));
142             } else {
143                 this.setExtendedState(JFrame.NORMAL);
144                 this.setLocationRelativeTo(null);
145                 Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
146                 int x = (int) ((dimension.getWidth() - Main.WIDTH) / 2);
147                 int y = (int) ((dimension.getHeight() - Main.HEIGHT) / 2);
148                 this.setBounds(x, y, Main.WIDTH, Main.HEIGHT);
149                 resize_btn.setIcon(new ImageIcon(resizeUp_image));
150             }
151             Main.maximized = !Main.maximized;
152         });
153 
154         minimize_btn.addChangeListener(evt -> {
155             if (minimize_btn.getModel().isPressed()) {
156                 this.setState(JFrame.ICONIFIED);
157                 minimize_btn.setForeground(Colors.primaryColor);
158             } else if (minimize_btn.getModel().isRollover()) {
159                 minimize_btn.setForeground(Colors.secondaryColor);
160             } else {
161                 minimize_btn.setForeground(Colors.primaryColor);
162             }
163         });
164 
165         backToMenu_btn = new JButton();
166         backToMenu_btn.setText("Back to Menu ");
167         backToMenu_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
168         backToMenu_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
169         backToMenu_btn.setFocusPainted(false);
170         backToMenu_btn.setContentAreaFilled(false);
171         backToMenu_btn.setOpaque(true);
172         backToMenu_btn.setBorder(null);
173         backToMenu_btn.addChangeListener(evt -> {
174             if (backToMenu_btn.getModel().isPressed()) {
175                 backToMenu_btn.setForeground(Colors.bgColor);
```

```
            } else if (backToMenu_btn.getModel().isRollover()) {
                backToMenu_btn.setForeground(Colors.accentColor);
            } else {
                backToMenu_btn.setForeground(Colors.bgColor);
            }
        });
        backToMenu_btn.addActionListener(e -> {
            this.setVisible(false);
            this.dispose();
            grantAccess = true;
            Main.changeFrame(1);
        });

        option1_btn = new JButton();
        option1_btn.setText(Topics[0] + " ");
        option1_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
        option1_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
        option1_btn.setFocusPainted(false);
        option1_btn.setBounds(0, 0, 500, 500);
        option1_btn.setContentAreaFilled(false);
        option1_btn.setOpaque(true);
        option1_btn.setBorder(null);
        option1_btn.addChangeListener(evt -> {
            if (option1_btn.getModel().isPressed()) {
                option1_btn.setForeground(Colors.accentColor);
            } else if (option1_btn.getModel().isRollover()) {
                option1_btn.setForeground(Colors.accentColor);
            } else {
                option1_btn.setForeground(Colors.primaryColor);
            }
        });

        option1_btn.addActionListener(e -> {
            grantAccess = true;
            currentTopic = Topics[0];
            this.setVisible(false);
            this.dispose();
            grantAccess = true;
            Main.changeFrame(6);
        });

        option2_btn = new JButton();
        option2_btn.setText(Topics[1] + " ");
        option2_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
        option2_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
        option2_btn.setFocusPainted(false);
        option2_btn.setContentAreaFilled(false);
        option2_btn.setOpaque(true);
        option2_btn.setBorder(null);
        option2_btn.addChangeListener(evt -> {
            if (option2_btn.getModel().isPressed()) {
                option2_btn.setForeground(Colors.accentColor);
            } else if (option2_btn.getModel().isRollover()) {
                option2_btn.setForeground(Colors.accentColor);
            } else {
                option2_btn.setForeground(Colors.primaryColor);
            }
        });
        option2_btn.addActionListener(e -> {
```

```
            grantAccess = true;
            currentTopic = Topics[1];
            this.setVisible(false);
            this.dispose();
            grantAccess = true;
            Main.changeFrame(6);
        });

        option3_btn = new JButton();
        option3_btn.setText(Topics[2] + " ");
        option3_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
        option3_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
        option3_btn.setFocusPainted(false);
        option3_btn.setContentAreaFilled(false);
        option3_btn.setOpaque(true);
        option3_btn.setBorder(null);
        option3_btn.addChangeListener(evt -> {
            if (option3_btn.getModel().isPressed()) {
                option3_btn.setForeground(Colors.accentColor);
            } else if (option3_btn.getModel().isRollover()) {
                option3_btn.setForeground(Colors.accentColor);
            } else {
                option3_btn.setForeground(Colors.primaryColor);
            }
        });
        option3_btn.addActionListener(e -> {
            grantAccess = true;
            currentTopic = Topics[2];
            this.setVisible(false);
            this.dispose();
            grantAccess = true;
            Main.changeFrame(6);
        });

        option4_btn = new JButton();
        option4_btn.setText(Topics[3] + " ");
        option4_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
        option4_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
        option4_btn.setFont(buttonFont.deriveFont(44f));
        option4_btn.setFocusPainted(false);
        option4_btn.setContentAreaFilled(false);
        option4_btn.setOpaque(true);
        option4_btn.setBorder(null);
        option4_btn.addChangeListener(evt -> {
            if (option4_btn.getModel().isPressed()) {
                option4_btn.setForeground(Colors.accentColor);
            } else if (option4_btn.getModel().isRollover()) {
                option4_btn.setForeground(Colors.accentColor);
            } else {
                option4_btn.setForeground(Colors.primaryColor);
            }
        });
        option4_btn.addActionListener(e -> {
            grantAccess = true;
            currentTopic = Topics[3];
            this.setVisible(false);
            this.dispose();
            grantAccess = true;
            Main.changeFrame(6);
```

```
294         });
295     }
296 
297 }
```

Listing 7: Main Java file

```
1  package org.howmuch;
2  
3  import javax.swing.*;
4  import java.awt.*;
5  import java.awt.event.ComponentAdapter;
6  import java.awt.event.ComponentEvent;
7  
8  import static org.howmuch.Main.*;
9  
10 public class GameOverFrame extends JFrame {
11     BackgroundPanel backgroundPanel;
12     JButton backtoTopic_btn;
13     JLabel score_lbl;
14 
15     GameOverFrame() {
16         backgroundPanel = new BackgroundPanel();
17 
18         this.setTitle("How Much? ");
19         if (maximized) {
20             this.setExtendedState(MAXIMIZED_BOTH);
21         } else {
22             this.setPreferredSize(new Dimension(Main.WIDTH, Main.HEIGHT));
23         }
24         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
25         this.setResizable(true);
26         this.setUndecorated(true);
27         this.setMinimumSize(new Dimension(1280, 720));
28 
29         createFonts();
30         createBasicButtonPanel();
31         createLabels();
32         createButtons();
33         reassignColors();
34         reassignBounds();
35 
36         this.addComponentListener(new ComponentAdapter() {
37             @Override
38             public void componentResized(ComponentEvent e) {
39                 reassignBounds();
40                 repaint();
41             }
42         });
43 
44         this.add(score_lbl);
45         this.add(backtoTopic_btn);
46         this.add(basicButtons_pnl);
47         this.add(backgroundPanel);
48         this.pack();
49         this.setLocationRelativeTo(null);
50         this.setVisible(true);
51     }
52 
```

```
53      private void reassignBounds() {
54          Dimension screenSize = this.getSize();
55
56          // The back to menu mode label
57          backtoTopic_btn.setBounds((int) (0.001 * screenSize.getWidth()), (int)
    (0.80 * screenSize.getHeight()),
58                  (int) (0.20 * screenSize.getWidth()), (int) (0.07 * screenSize.
    getHeight()));
59          backtoTopic_btn.setFont(buttonFont.deriveFont((float) (0.06 * getHeight())
    ));
60
61          // The Entire basic button panel for closing minimizing and stuff
62          basicButtons_pnl.setBounds(this.getWidth() - (exit_btn.getWidth() * 3) -
    40, 10, exit_btn.getWidth() * 3 + 35,
63                  exit_btn.getHeight());
64
65          // Score Label
66          score_lbl.setBounds((int) (0.76 * screenSize.getWidth()), (int) (0.80 *
    screenSize.getHeight()),
67                  (int) (0.31 * screenSize.getWidth()), (int) (0.13 * screenSize.
    getHeight()));
68          score_lbl.setFont(buttonFont.deriveFont((float) (0.14 * getHeight())));
69      }
70
71      private void reassignColors() {
72          Colors.reassignColors();
73          if (GameFrame.gameWon) {
74              if (Colors.DarkMode) {
75                  backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java
    /How Much/src/main/resources/images/game won over dark.png");
76              } else {
77                  backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java
    /How Much/src/main/resources/images/game won over.png");
78              }
79          } else {
80              if (Colors.DarkMode) {
81                  backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java
    /How Much/src/main/resources/images/game over dark.png");
82              } else {
83                  backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java
    /How Much/src/main/resources/images/game over.png");
84              }
85          }
86          backtoTopic_btn.setBackground(Colors.primaryColor);
87          backtoTopic_btn.setForeground(Colors.bgColor);
88
89          score_lbl.setBackground(Colors.bgColor);
90          score_lbl.setForeground(Colors.accentColor);
91
92          basicButtons_pnl.setBackground(Colors.bgColor);
93          exit_btn.setBackground(Colors.bgColor);
94          resize_btn.setBackground(Colors.bgColor);
95          minimize_btn.setBackground(Colors.bgColor);
96      }
97
98      private void createLabels() {
99          score_lbl = new JLabel();
100         score_lbl.setText(String.valueOf(DataBaseManager.currentScore));
101         score_lbl.setAlignmentY(Box.CENTER_ALIGNMENT);
```

```
102         score_lbl.setAlignmentX(Box.CENTER_ALIGNMENT);
103         score_lbl.setOpaque(true);
104         score_lbl.setBorder(null);
105     }
106 
107     private void createButtons() {
108         backtoTopic_btn = new JButton();
109         backtoTopic_btn.setText("Try Again");
110         backtoTopic_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
111         backtoTopic_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
112         backtoTopic_btn.setFocusPainted(false);
113         backtoTopic_btn.setContentAreaFilled(false);
114         backtoTopic_btn.setOpaque(true);
115         backtoTopic_btn.setBorder(null);
116         backtoTopic_btn.addChangeListener(evt -> {
117             if (backtoTopic_btn.getModel().isPressed()) {
118                 backtoTopic_btn.setForeground(Colors.bgColor);
119             } else if (backtoTopic_btn.getModel().isRollover()) {
120                 backtoTopic_btn.setForeground(Colors.accentColor);
121             } else {
122                 backtoTopic_btn.setForeground(Colors.bgColor);
123             }
124         });
125         backtoTopic_btn.addActionListener(e -> {
126             this.setVisible(false);
127             this.dispose();
128             grantAccess = true;
129             Main.changeFrame(2);
130         });
131 
132         // Removing Change and Action Listeners.
133         GameFrame.removeAllChangeAndActionListenersFromBasicButtons();
134 
135         exit_btn.addChangeListener(evt -> {
136             if (exit_btn.getModel().isPressed()) {
137                 exit_btn.setForeground(Colors.primaryColor);
138                 Main.changeFrame(0);
139             } else if (exit_btn.getModel().isRollover()) {
140                 exit_btn.setForeground(Colors.secondaryColor);
141             } else {
142                 exit_btn.setForeground(Colors.primaryColor);
143             }
144         });
145         resize_btn.addActionListener(e -> {
146             if (!Main.maximized) {
147                 this.setExtendedState(MAXIMIZED_BOTH);
148                 resize_btn.setIcon(new ImageIcon(resizeDown_image));
149             } else {
150                 this.setExtendedState(JFrame.NORMAL);
151                 this.setLocationRelativeTo(null);
152                 Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
153                 int x = (int) ((dimension.getWidth() - Main.WIDTH) / 2);
154                 int y = (int) ((dimension.getHeight() - Main.HEIGHT) / 2);
155                 this.setBounds(x, y, Main.WIDTH, Main.HEIGHT);
156                 resize_btn.setIcon(new ImageIcon(resizeUp_image));
157             }
158             Main.maximized = !Main.maximized;
159         });
160 
```

```
        minimize_btn.addChangeListener(evt -> {
            if (minimize_btn.getModel().isPressed()) {
                this.setState(JFrame.ICONIFIED);
                minimize_btn.setForeground(Colors.primaryColor);
            } else if (minimize_btn.getModel().isRollover()) {
                minimize_btn.setForeground(Colors.secondaryColor);
            } else {
                minimize_btn.setForeground(Colors.primaryColor);
            }
        });
    }
}
```

Listing 8: Main Java file

```
/*
 * Class that does everything that we wanna do with mongodb. Things like inserting
    , deleting, creating and fetching data.
 */

package org.howmuch;

import com.mongodb.MongoClient;
import com.mongodb.client.*;
import org.bson.Document;

public class MongoManager {
    static MongoDatabase database;
    public static String MONGO_DATABASE_NAME = "HowMuch";
    public static int MONGO_PORT_NO = 27017;
    public static String MONGO_HOST = "localhost";

    public static String[] fetchDataFromMongo(String currentTopic, int randomIndex
    ) {
        MongoCollection<org.bson.Document> collection = database.getCollection(
    currentTopic);
        FindIterable<Document> iterDoc = collection.find();
        int i = 0;
        for (Document document : iterDoc) {
            System.out.println(document);
            if (i == randomIndex) {
                return new String[] { (String) document.get("Name"), document.
    getString("Price"),
                        document.getString("Image") };
            }
            i++;
        }
        return new String[] { "Sadly Not Found", "Sadly Not Found", "Sadly Not
    Found" };

    }

    public static boolean establishConnectionWithMongo() {
        // Creating a MongoDB client
        try {
            MongoClient mongoClient = new MongoClient(MONGO_HOST, MONGO_PORT_NO);
            // Connecting to the database
            database = mongoClient.getDatabase(MONGO_DATABASE_NAME);
            System.out.println("Connected Successfully to mongoDb");
```

```
            return true;

        } catch (Exception e) {
            System.out.println("Couldnt establish connection due to some reason");
            System.out.println(e.getMessage());
            return false;
        }
    }

    public static void addDataToMongo(String Topic, String[] data) {
        try {
            MongoClient mongoClient = new MongoClient(MONGO_HOST, MONGO_PORT_NO);
            // Connecting to the database
            database = mongoClient.getDatabase(MONGO_DATABASE_NAME);
            Topic = Topic.substring(0, 1).toUpperCase() + Topic.substring(1);
            MongoCollection<Document> collection = database.getCollection(Topic);
            Document dataDocToAdd = new Document();
            dataDocToAdd.append("Name", data[0]);
            dataDocToAdd.append("Price", data[1]);
            dataDocToAdd.append("Image", data[2]);
            collection.insertOne(dataDocToAdd);
            System.out.println(data[0]);
            System.out.println("\n\nAdded record to mongo---------\n\n");
        } catch (Exception e) {
            System.out.println("Couldnt add data");
        }
    }

    public static void clearMongoDb() {
        MongoCollection<Document> collection = database.getCollection(Main.Topics
    [0]);
        collection.drop();
        collection = database.getCollection(Main.Topics[1]);
        collection.drop();
        collection = database.getCollection(Main.Topics[2]);
        collection.drop();
        collection = database.getCollection(Main.Topics[3]);
        collection.drop();
    }
}
```

Listing 9: Main Java file

```
/*
 * An Important class, As it controls a lot of the important variables, and all
    interactions with the Local CSV File databases.
 */

package org.howmuch;

import com.mongodb.client.MongoDatabase;
import com.opencsv.CSVReader;
import com.opencsv.CSVWriter;
import org.apache.commons.io.FileUtils;

import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
```

```
16 import java.util.Arrays;
17 import java.util.List;
18 import java.util.Objects;
19
20 public class DataBaseManager {
21
22     public static String LOCAL_DATAFOLDER = "/run/media/krishnaraj/Programs/Java/
    How Much/src/main/resources/data";
23     public static String LOCAL_CSV_FOLDER = "/run/media/krishnaraj/Programs/Java/
    How Much/src/main/resources/data/csvs";
24     public static String LOCAL_IMG_FOLDER = "/run/media/krishnaraj/Programs/Java/
    How Much/src/main/resources/data/images";
25     public static String LOCAL_BACKUP_DATAFOLDER = "/run/media/krishnaraj/Programs
    /Java/How Much/src/main/resources/data_backup";
26     public static String LOCAL_BACKUP_CSV_FOLDER = "/run/media/krishnaraj/Programs
    /Java/How Much/src/main/resources/data_backup/csvs";
27     public static String LOCAL_BACKUP_IMG_FOLDER = "/run/media/krishnaraj/Programs
    /Java/How Much/src/main/resources/data_backup/images";
28     public static String USERDATA_FILEPATH = "/run/media/krishnaraj/Programs/Java/
    How Much/src/main/resources/data/user_details.csv";
29     public static String BACKUP_USERDATA_FILEPATH = "/run/media/krishnaraj/
    Programs/Java/How Much/src/main/resources/data_backup/user_details.csv";
30     public static String LOCAL_DATEFILE = "/run/media/krishnaraj/Programs/Java/How
     Much/src/main/resources/data/dateUpdated.txt";
31     public static String LOCAL_MONGODATEFILE = "/run/media/krishnaraj/Programs/
    Java/How Much/src/main/resources/data/MongoDateUpdated.txt";
32     public static String LOCAL_BACKUP_DATEFILE = "/run/media/krishnaraj/Programs/
    Java/How Much/src/main/resources/data_backup/dateUpdated.txt";
33
34     static String currentUsername = "guest";
35     static int USER_INDEX = -1;
36     static String currentPassword = "guest";
37     static int currentScore = 0;
38
39     /**
40      * Brutally Clear the images and csv in the local Database and start fresh
    with
41      * only files.
42      **/
43     public static void clearLocalDatabase() {
44         try {
45             // Delete all pre existing images
46             File data_deleter = new File(LOCAL_IMG_FOLDER);
47             listFilesForFolder(data_deleter);
48             for (File subfile : Objects.requireNonNull(data_deleter.listFiles()))
    {
49                 if (subfile.isDirectory()) {
50                     for (File f : Objects.requireNonNull(subfile.listFiles())) {
51                         f.delete();
52                     }
53                 }
54             }
55
56             // Also clear the csv files.
57             data_deleter = new File(LOCAL_CSV_FOLDER);
58             listFilesForFolder(data_deleter);
59             for (File subfile : Objects.requireNonNull(data_deleter.listFiles()))
    {
60                 subfile.delete();
```

```
61              }
62
63              // Recreate them.
64              File createfiles = new File(LOCAL_BACKUP_CSV_FOLDER + "/" + Main.
    Topics[0].toLowerCase() + ".csv");
65              createfiles.createNewFile();
66              createfiles = new File(LOCAL_BACKUP_CSV_FOLDER + "/" + Main.Topics[1].
    toLowerCase() + ".csv");
67              createfiles.createNewFile();
68              createfiles = new File(LOCAL_BACKUP_CSV_FOLDER + "/" + Main.Topics[2].
    toLowerCase() + ".csv");
69              createfiles.createNewFile();
70              createfiles = new File(LOCAL_BACKUP_CSV_FOLDER + "/" + Main.Topics[3].
    toLowerCase() + ".csv");
71              createfiles.createNewFile();
72          } catch (Exception e) {
73              System.out.println("Some io excepition occured");
74          }
75      }
76
77      /*
78       * Simply displays every file in a directory
79       */
80      public static void listFilesForFolder(final File folder) {
81          Arrays.stream(folder.listFiles()).forEach(fileEntry -> {
82              if (fileEntry.isDirectory()) {
83                  listFilesForFolder(fileEntry);
84              } else {
85                  System.out.println(fileEntry.getName());
86                  System.out.println(fileEntry.getPath());
87              }
88          });
89      }
90
91      /*
92       * Adds a new user to the local CSV Database. Creates that file if it doesnt
93       * exist.
94       */
95      public static void addNewUser() {
96          System.out.println("gonna add new user");
97          File userDatafile = new File(USERDATA_FILEPATH);
98
99          try (CSVReader reader = new CSVReader(new FileReader(userDatafile), ','))
    {
100             List<String[]> csvBody = reader.readAll();
101             USER_INDEX = csvBody.size();
102         } catch (IOException e) {
103             throw new RuntimeException(e);
104         }
105
106         // append the new user to the login file.
107         try (FileWriter userDataFileWriter = new FileWriter(userDatafile, true)) {
108
109             // create CSVWriter object filewriter object as parameter
110             try (CSVWriter writer = new CSVWriter(userDataFileWriter, CSVWriter.
    DEFAULT_SEPARATOR,
111                     CSVWriter.NO_QUOTE_CHARACTER, CSVWriter.
    DEFAULT_ESCAPE_CHARACTER, CSVWriter.DEFAULT_LINE_END)) {
112
```

```
113                 String[] data = { currentUsername, currentPassword, String.valueOf
    (currentScore) };
114                 writer.writeNext(data);
115                 System.out.println("added new user");
116             }
117 
118         } catch (IOException e) {
119             System.out.println("Cant open user data file. ");
120         }
121     }
122 
123     public static void addDataToCSV(String filePath, String[] data) {
124         File userDatafile = new File(filePath);
125 
126         // append the new user to the login file.
127         try (FileWriter userDataFileWriter = new FileWriter(userDatafile, true)) {
128 
129             // create CSVWriter object filewriter object as parameter
130             try (CSVWriter writer = new CSVWriter(userDataFileWriter, CSVWriter.
    DEFAULT_SEPARATOR,
131                     CSVWriter.NO_QUOTE_CHARACTER, CSVWriter.
    DEFAULT_ESCAPE_CHARACTER, CSVWriter.DEFAULT_LINE_END)) {
132                 // System.out.println(Arrays.toString(data));
133                 writer.writeNext(data);
134             }
135 
136         } catch (IOException e) {
137             System.out.println("Cant open user data file. ");
138         }
139     }
140 
141     public static boolean doesUsernameExist(String username) {
142         File inputFile = new File(USERDATA_FILEPATH);
143         try (CSVReader reader = new CSVReader(new FileReader(inputFile), ',')) {
144             List<String[]> csvBody = reader.readAll();
145             for (String[] s : csvBody) {
146                 if (s[0].equals(username)) {
147                     System.out.println("User Already Exists");
148                     return true;
149                 }
150             }
151         } catch (IOException e) {
152             System.out.println("couldnt create csvreader in username exists
    checker method. ");
153         }
154         return false;
155     }
156 
157     public static boolean doesPasswordMatch(String username, String password) {
158         File inputFile = new File(USERDATA_FILEPATH);
159         try (CSVReader reader = new CSVReader(new FileReader(inputFile), ',')) {
160             List<String[]> csvBody = reader.readAll();
161             for (int i = 0; i < csvBody.size(); i++) {
162                 String[] s = csvBody.get(i);
163                 if (s[0].equals(username)) {
164                     System.out.println("User Found");
165                     if (s[1].equals(password)) {
166                         System.out.println("Password Matches");
167                         USER_INDEX = i;
```

```
168                         return true;
169                     } else
170                         return false;
171                 }
172             }
173         } catch (IOException e) {
174             System.out.println("couldnt create csvreader in password matching
    method");
175         }
176         return false;
177     }
178 
179     public static List<String[]> getStoredUserScores() {
180         File inputFile = new File(USERDATA_FILEPATH);
181         List<String[]> csvBody = null;
182         try (CSVReader reader = new CSVReader(new FileReader(inputFile), ',')) {
183             csvBody = reader.readAll();
184             return csvBody;
185         } catch (IOException e) {
186             System.out.println("couldnt create csvreader in userscore method");
187         }
188         return csvBody;
189     }
190 
191     public static void updateUserScore() {
192 
193         File inputFile = new File(USERDATA_FILEPATH);
194 
195         List<String[]> csvBody;
196         try (CSVReader reader = new CSVReader(new FileReader(inputFile), ',')) {
197             csvBody = reader.readAll();
198             csvBody.get(USER_INDEX)[2] = String.valueOf(currentScore);
199         } catch (IOException e) {
200             throw new RuntimeException(e);
201         }
202 
203         try (CSVWriter writer = new CSVWriter(new FileWriter(inputFile), ',')) {
204             writer.writeAll(csvBody);
205             writer.flush();
206         } catch (IOException e) {
207             throw new RuntimeException(e);
208         }
209     }
210 
211     public static void createLocalDatabaseBackupOfUsers() {
212         System.out.println("---------------CREATING LOCAL DATABASE BACKUP of the
    user file------------");
213         try {
214             File sourceDirectory = new File(USERDATA_FILEPATH);
215             File destinationDirectory = new File(BACKUP_USERDATA_FILEPATH);
216             FileUtils.copyFile(sourceDirectory, destinationDirectory);
217         } catch (IOException e) {
218             throw new RuntimeException(e);
219         }
220     }
221 
222     public static void createLocalDatabaseBackup() {
223         try {
224             System.out.println("---------------CREATING LOCAL DATABASE BACKUP
```

```
    ------------");
225             // Delete all pre existing images
226             File data_deleter = new File(LOCAL_BACKUP_IMG_FOLDER);
227             // listFilesForFolder(data_deleter);
228             for (File subfile : Objects.requireNonNull(data_deleter.listFiles()))
    {
229                 if (subfile.isDirectory()) {
230                     for (File f : Objects.requireNonNull(subfile.listFiles())) {
231                         f.delete();
232                     }
233                 }
234             }
235             data_deleter = new File(LOCAL_BACKUP_CSV_FOLDER);
236             for (File subfile : Objects.requireNonNull(data_deleter.listFiles()))
    {
237                 subfile.delete();
238             }
239             File createfiles = new File(LOCAL_BACKUP_CSV_FOLDER + "/" + Main.
    Topics[0].toLowerCase() + ".csv");
240             createfiles.createNewFile();
241             createfiles = new File(LOCAL_BACKUP_CSV_FOLDER + "/" + Main.Topics[1].
    toLowerCase() + ".csv");
242             createfiles.createNewFile();
243             createfiles = new File(LOCAL_BACKUP_CSV_FOLDER + "/" + Main.Topics[2].
    toLowerCase() + ".csv");
244             createfiles.createNewFile();
245             createfiles = new File(LOCAL_BACKUP_CSV_FOLDER + "/" + Main.Topics[3].
    toLowerCase() + ".csv");
246             createfiles.createNewFile();
247         } catch (Exception e) {
248             System.out.println("Some io excepition occured");
249         }
250
251         try {
252             File sourceDirectory = new File(LOCAL_DATAFOLDER);
253             File destinationDirectory = new File(LOCAL_BACKUP_DATAFOLDER);
254             FileUtils.copyDirectory(sourceDirectory, destinationDirectory);
255
256         } catch (IOException e) {
257             throw new RuntimeException(e);
258         }
259     }
260
261     public static String[] readFromLocalDatabase(String Topic, int index) {
262         File inputFile;
263         if (Main.isLocalDatabaseUpToDate) {
264             System.out.println("running from the local database");
265             inputFile = new File(LOCAL_CSV_FOLDER + '/' + Topic.toLowerCase() + ".
    csv");
266             try (CSVReader reader = new CSVReader(new FileReader(inputFile), ','))
     {
267                 List<String[]> csvBody = reader.readAll();
268                 if (index > csvBody.size()) {
269                     return csvBody.get(csvBody.size() - 1);
270                 }
271                 return csvBody.get(index);
272             } catch (IOException e) {
273                 throw new RuntimeException(e);
274             }
```

```
275
276         } else {
277             System.out.println("running from the backup local database");
278             inputFile = new File(LOCAL_BACKUP_CSV_FOLDER + '/' + Topic.toLowerCase
    () + ".csv");
279             try (CSVReader reader = new CSVReader(new FileReader(inputFile), ','))
     {
280                 List<String[]> csvBody = reader.readAll();
281                 if (index > csvBody.size()) {
282                     String[] s;
283                     s = csvBody.get(csvBody.size() - 1);
284                     s[2] = s[2].replace("/data/", "/data_backup/");
285                     return s;
286                 }
287                 String[] s;
288                 s = csvBody.get(index);
289                 s[2] = s[2].replace("/data/", "/data_backup/");
290                 return s;
291             } catch (IOException e) {
292                 throw new RuntimeException(e);
293             }
294         }
295
296     }
297
298     public static int findLength(String Topic) {
299         File inputFile;
300         if (Main.isLocalDatabaseUpToDate) {
301             inputFile = new File(LOCAL_CSV_FOLDER + '/' + Topic.toLowerCase() + ".
    csv");
302         } else {
303             inputFile = new File(LOCAL_BACKUP_CSV_FOLDER + '/' + Topic.toLowerCase
    () + ".csv");
304         }
305         try (CSVReader reader = new CSVReader(new FileReader(inputFile), ',')) {
306             List<String[]> csvBody = reader.readAll();
307             return csvBody.size();
308         } catch (IOException e) {
309             throw new RuntimeException(e);
310         }
311     }
312 }
```

Listing 10: Main Java file

```
1  package org.howmuch;
2
3  import javax.swing.*;
4  import java.awt.*;
5  import java.awt.event.ComponentAdapter;
6  import java.awt.event.ComponentEvent;
7  import java.util.ArrayList;
8  import java.util.Arrays;
9  import java.util.Collections;
10
11 import static org.howmuch.Main.*;
12
13 public class HighscoreFrame extends JFrame {
14     BackgroundPanel backgroundPanel;
```

```
    JButton backToMenu_btn;
    JTextArea highScores_txtArea;

    HighscoreFrame() {
        backgroundPanel = new BackgroundPanel();

        this.setTitle("How Much? ");
        if (maximized) {
            this.setExtendedState(MAXIMIZED_BOTH);
        } else {
            this.setPreferredSize(new Dimension(Main.WIDTH, Main.HEIGHT));
        }
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setResizable(true);
        this.setUndecorated(true);
        this.setMinimumSize(new Dimension(1280, 720));

        createFonts();
        createBasicButtonPanel();
        createButtons();
        createLabels();
        reassignColors();
        reassignBounds();

        this.addComponentListener(new ComponentAdapter() {
            @Override
            public void componentResized(ComponentEvent e) {
                reassignBounds();
                repaint();
            }
        });

        this.add(highScores_txtArea);
        this.add(backToMenu_btn);
        this.add(basicButtons_pnl);
        this.add(backgroundPanel);
        this.pack();
        this.setLocationRelativeTo(null);
        this.setVisible(true);
    }

    private void createLabels() {
        highScores_txtArea = new JTextArea();
        int peopleCount = 0;
        java.util.List<String[]> userData = DataBaseManager.getStoredUserScores();
        for (int i = 0; i < userData.size(); i++) {
            System.out.println(Integer.parseInt(userData.get(i)[2]));
        }
        ArrayList<Integer> scores = new ArrayList<>();
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < userData.size(); i++) {
            System.out.println(Integer.parseInt(userData.get(i)[2]));
            scores.add(Integer.parseInt(userData.get(i)[2]));
        }
        scores.sort(Collections.reverseOrder());

        for (int i = 0; i < scores.size(); i++) {
            System.out.println(Integer.parseInt(String.valueOf(scores.get(i))));
        }
```

```
74          System.out.println("fianls");
75          for (int i = 0; i < scores.size(); i++) {
76              for (int j = 0; j < userData.size(); j++) {
77                  if (Integer.valueOf(userData.get(j)[2]).equals(scores.get(i))) {
78                      System.out.println(Arrays.toString(userData.get(j)));
79                      sb.append(userData.get(j)[0] + "    -    " + userData.get(j)[2]
    + "\n");
80                      peopleCount++;
81                      if (peopleCount == 5) {
82                          break;
83                      }
84                  }
85              }
86              if (peopleCount == 5) {
87                  break;
88              }
89          }
90          highScores_txtArea.setText(String.valueOf(sb));
91          highScores_txtArea.setAlignmentY(Box.CENTER_ALIGNMENT);
92          highScores_txtArea.setAlignmentX(Box.LEFT_ALIGNMENT);
93          highScores_txtArea.setOpaque(true);
94          highScores_txtArea.setBorder(null);
95          highScores_txtArea.setLineWrap(true);
96      }
97
98      private void reassignBounds() {
99          Dimension screenSize = this.getSize();
100
101         // The back to menu mode label
102         backToMenu_btn.setBounds((int) (0.015 * screenSize.getWidth()), (int)
    (0.80 * screenSize.getHeight()),
103                 (int) (0.20 * screenSize.getWidth()), (int) (0.07 * screenSize.
    getHeight()));
104         backToMenu_btn.setFont(buttonFont.deriveFont((float) (0.05 * getHeight()))
    );
105
106         // The Entire basic button panel for closing minimizing and stuff
107         basicButtons_pnl.setBounds(this.getWidth() - (exit_btn.getWidth() * 3) -
    40, 10, exit_btn.getWidth() * 3 + 35,
108                 exit_btn.getHeight());
109         highScores_txtArea.setBounds((int) (0.60 * screenSize.getWidth()), (int)
    (0.38 * screenSize.getHeight()),
110                 (int) (0.60 * screenSize.getWidth()), 700);
111         highScores_txtArea.setFont(textFont.deriveFont(44f));
112
113     }
114
115     private void reassignColors() {
116         Colors.reassignColors();
117         if (Colors.DarkMode) {
118             backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java/How
     Much/src/main/resources/images/highscore dark.png");
119         } else {
120             backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java/How
     Much/src/main/resources/images/highscore.png");
121         }
122         backToMenu_btn.setBackground(Colors.primaryColor);
123         backToMenu_btn.setForeground(Colors.bgColor);
124
```

```
125         basicButtons_pnl.setBackground(Colors.bgColor);
126         exit_btn.setBackground(Colors.bgColor);
127         resize_btn.setBackground(Colors.bgColor);
128         minimize_btn.setBackground(Colors.bgColor);
129         highScores_txtArea.setBackground(Colors.bgColor);
130         highScores_txtArea.setForeground(Colors.primaryColor);
131     }
132 
133     private void createButtons() {
134         backToMenu_btn = new JButton();
135         backToMenu_btn.setText("Back to Menu");
136         backToMenu_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
137         backToMenu_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
138         backToMenu_btn.setFocusPainted(false);
139         backToMenu_btn.setContentAreaFilled(false);
140         backToMenu_btn.setOpaque(true);
141         backToMenu_btn.setBorder(null);
142         backToMenu_btn.addChangeListener(evt -> {
143             if (backToMenu_btn.getModel().isPressed()) {
144                 backToMenu_btn.setForeground(Colors.bgColor);
145             } else if (backToMenu_btn.getModel().isRollover()) {
146                 backToMenu_btn.setForeground(Colors.accentColor);
147             } else {
148                 backToMenu_btn.setForeground(Colors.bgColor);
149             }
150         });
151         backToMenu_btn.addActionListener(e -> {
152             this.setVisible(false);
153             this.dispose();
154             grantAccess = true;
155             Main.changeFrame(1);
156         });
157 
158         // Removing Change and Action Listeners.
159         GameFrame.removeAllChangeAndActionListenersFromBasicButtons();
160 
161         exit_btn.addChangeListener(evt -> {
162             if (exit_btn.getModel().isPressed()) {
163                 exit_btn.setForeground(Colors.primaryColor);
164                 Main.changeFrame(0);
165             } else if (exit_btn.getModel().isRollover()) {
166                 exit_btn.setForeground(Colors.secondaryColor);
167             } else {
168                 exit_btn.setForeground(Colors.primaryColor);
169             }
170         });
171         resize_btn.addActionListener(e -> {
172             if (!Main.maximized) {
173                 this.setExtendedState(MAXIMIZED_BOTH);
174                 resize_btn.setIcon(new ImageIcon(resizeDown_image));
175             } else {
176                 this.setExtendedState(JFrame.NORMAL);
177                 this.setLocationRelativeTo(null);
178                 Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
179                 int x = (int) ((dimension.getWidth() - Main.WIDTH) / 2);
180                 int y = (int) ((dimension.getHeight() - Main.HEIGHT) / 2);
181                 this.setBounds(x, y, Main.WIDTH, Main.HEIGHT);
182                 resize_btn.setIcon(new ImageIcon(resizeUp_image));
183             }
```

```
            Main.maximized = !Main.maximized;
        });

        minimize_btn.addChangeListener(evt -> {
            if (minimize_btn.getModel().isPressed()) {
                this.setState(JFrame.ICONIFIED);
                minimize_btn.setForeground(Colors.primaryColor);
            } else if (minimize_btn.getModel().isRollover()) {
                minimize_btn.setForeground(Colors.secondaryColor);
            } else {
                minimize_btn.setForeground(Colors.primaryColor);
            }
        });
    }
}
```

Listing 11: Main Java file

```
package org.howmuch;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;

import static org.howmuch.Main.*;

public class HelpFrame extends JFrame {
    BackgroundPanel backgroundPanel;
    JButton backToMenu_btn;

    HelpFrame() {
        backgroundPanel = new BackgroundPanel();

        this.setTitle("How Much? ");
        if (maximized) {
            this.setExtendedState(MAXIMIZED_BOTH);
        } else {
            this.setPreferredSize(new Dimension(Main.WIDTH, Main.HEIGHT));
        }
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setResizable(true);
        this.setUndecorated(true);
        this.setMinimumSize(new Dimension(1280, 720));

        createFonts();
        createBasicButtonPanel();
        createButtons();
        reassignColors();
        reassignBounds();

        this.addComponentListener(new ComponentAdapter() {
            @Override
            public void componentResized(ComponentEvent e) {
                reassignBounds();
                repaint();
            }
        });

```

```
42          this.add(backToMenu_btn);
43          this.add(basicButtons_pnl);
44          this.add(backgroundPanel);
45          this.pack();
46          this.setLocationRelativeTo(null);
47          this.setVisible(true);
48      }
49
50      private void reassignBounds() {
51          Dimension screenSize = this.getSize();
52
53          // The back to menu mode label
54          backToMenu_btn.setBounds((int) (0.015 * screenSize.getWidth()), (int)
    (0.80 * screenSize.getHeight()),
55                  (int) (0.20 * screenSize.getWidth()), (int) (0.07 * screenSize.
    getHeight()));
56          backToMenu_btn.setFont(buttonFont.deriveFont((float) (0.05 * getHeight()))
    );
57
58          // The Entire basic button panel for closing minimizing and stuff
59          basicButtons_pnl.setBounds(this.getWidth() - (exit_btn.getWidth() * 3) -
    40, 10, exit_btn.getWidth() * 3 + 35,
60                  exit_btn.getHeight());
61
62      }
63
64      private void reassignColors() {
65          Colors.reassignColors();
66          if (Colors.DarkMode) {
67              backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java/How
     Much/src/main/resources/images/help and credits dark.png");
68          } else {
69              backgroundPanel.setBackground("/run/media/krishnaraj/Programs/Java/How
     Much/src/main/resources/images/help and credits.png");
70          }
71          backToMenu_btn.setBackground(Colors.primaryColor);
72          backToMenu_btn.setForeground(Colors.bgColor);
73
74          basicButtons_pnl.setBackground(Colors.bgColor);
75          exit_btn.setBackground(Colors.bgColor);
76          resize_btn.setBackground(Colors.bgColor);
77          minimize_btn.setBackground(Colors.bgColor);
78      }
79
80      private void createButtons() {
81          backToMenu_btn = new JButton();
82          backToMenu_btn.setText("Back to Menu");
83          backToMenu_btn.setAlignmentY(Box.CENTER_ALIGNMENT);
84          backToMenu_btn.setAlignmentX(Box.LEFT_ALIGNMENT);
85          backToMenu_btn.setFocusPainted(false);
86          backToMenu_btn.setContentAreaFilled(false);
87          backToMenu_btn.setOpaque(true);
88          backToMenu_btn.setBorder(null);
89          backToMenu_btn.addChangeListener(evt -> {
90              if (backToMenu_btn.getModel().isPressed()) {
91                  backToMenu_btn.setForeground(Colors.bgColor);
92              } else if (backToMenu_btn.getModel().isRollover()) {
93                  backToMenu_btn.setForeground(Colors.accentColor);
94              } else {
```

```
                backToMenu_btn.setForeground(Colors.bgColor);
            }
        });
        backToMenu_btn.addActionListener(e -> {
            this.setVisible(false);
            this.dispose();
            grantAccess = true;
            Main.changeFrame(1);
        });

        // Removing Change and Action Listeners.
        GameFrame.removeAllChangeAndActionListenersFromBasicButtons();

        exit_btn.addChangeListener(evt -> {
            if (exit_btn.getModel().isPressed()) {
                exit_btn.setForeground(Colors.primaryColor);
                Main.changeFrame(0);
            } else if (exit_btn.getModel().isRollover()) {
                exit_btn.setForeground(Colors.secondaryColor);
            } else {
                exit_btn.setForeground(Colors.primaryColor);
            }
        });
        resize_btn.addActionListener(e -> {
            if (!Main.maximized) {
                this.setExtendedState(MAXIMIZED_BOTH);
                resize_btn.setIcon(new ImageIcon(resizeDown_image));
            } else {
                this.setExtendedState(JFrame.NORMAL);
                this.setLocationRelativeTo(null);
                Dimension dimension = Toolkit.getDefaultToolkit().getScreenSize();
                int x = (int) ((dimension.getWidth() - Main.WIDTH) / 2);
                int y = (int) ((dimension.getHeight() - Main.HEIGHT) / 2);
                this.setBounds(x, y, Main.WIDTH, Main.HEIGHT);
                resize_btn.setIcon(new ImageIcon(resizeUp_image));
            }
            Main.maximized = !Main.maximized;
        });

        minimize_btn.addChangeListener(evt -> {
            if (minimize_btn.getModel().isPressed()) {
                this.setState(JFrame.ICONIFIED);
                minimize_btn.setForeground(Colors.primaryColor);
            } else if (minimize_btn.getModel().isRollover()) {
                minimize_btn.setForeground(Colors.secondaryColor);
            } else {
                minimize_btn.setForeground(Colors.primaryColor);
            }
        });
    }
}
```

Listing 12: Main Java file

```
/*
 * The class that Performs the role of the heart of the code. It is what downloads
    the images from aamzon, and saves them.
 * It scraps them, and we get the html page from the website. From there we can
   get the parts that we need as xml and then by parsing that xml
```

```
 * We can then get exactly what we want.
 */

package org.howmuch;

import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.*;

import com.groupdocs.conversion.Converter;
import com.groupdocs.conversion.filetypes.ImageFileType;
import com.groupdocs.conversion.options.convert.ImageConvertOptions;
import org.apache.commons.io.FileExistsException;
import org.w3c.dom.*;

import javax.xml.parsers.*;
import java.io.*;

import com.gargoylesoftware.htmlunit.*;
import com.gargoylesoftware.htmlunit.html.*;
import org.xml.sax.SAXException;

public class AmazonScrapper {
    static Converter converter;
    static ImageConvertOptions options;
    static WebClient webClient;
    static DocumentBuilder builder;
    static DocumentBuilderFactory factory;
    public static HashMap<Integer, String[]> searchQueries_map = new HashMap<>();
    public static String AMAZON_PREFIX_URL = "https://www.amazon.in/s?k=";

    AmazonScrapper() {
        options = new ImageConvertOptions();
        options.setFormat(ImageFileType.Png);
        fillSearchQueries();

        factory = DocumentBuilderFactory.newInstance();
        try {
            builder = factory.newDocumentBuilder();
        } catch (ParserConfigurationException e) {
            throw new RuntimeException(e);
        }

        // Define and declare basic web browser
        webClient = new WebClient(BrowserVersion.CHROME);
        webClient.getOptions().setCssEnabled(false);
        webClient.getOptions().setThrowExceptionOnFailingStatusCode(false);
        webClient.getOptions().setJavaScriptEnabled(false);
        webClient.getOptions().setThrowExceptionOnScriptError(false);
        webClient.getOptions().setPrintContentOnFailingStatusCode(false);
    }

    public static void fillSearchQueries() {
        System.out.println(Arrays.toString(Main.Topics));
```

```
63
64 //         This is the final stuff here, but is commented out for quicker
       debugging.
65
66         searchQueries_map.put(0, new String[]{"Televisions", "Mobile Phones",
67         "Laptops", "Iphone", "Macbook", "Refrigerators", "Washing Machines", "
       Smart Watches", "Gaming Laptops", "Computer Accessories", "GPUs", "Tablets",
68         "Playstation", "Xbox"});
69         searchQueries_map.put(1, new String[]{"Mens TShirts", "Formal Suits", "
       Mens Casual Wear", "Womens Casual Wear", "Womens Formal Wear", "Kids Clothes",
70         "Makeup", "Beauty Products", "Analog Watches", "Earrings", "Necklaces",
71         "Jewellery", "Branded Clothes", "Gold Jewellery", "Shoes"});
72         searchQueries_map.put(2, new String[]{"Furniture", "Tape", "Stationary",
       "Cutlery", "Kitchen Products", "Toothpaste", "Chocolates", "Soaps", "Water
       Bottles", "Carpets", "Sofa Sets", "Tables and Desks", "Cleaning Products"});
73         searchQueries_map.put(3, new String[]{"Gifts", "Car Appliances", "Diwali
       Lights", "Decoration", "Birthday Decor", "Lenses"});
74 //
75 //         searchQueries_map.put(0, new String[] { "8k OLED Televisions" });
76 //         searchQueries_map.put(1, new String[] { "Kurti", "Womens Dresses" });
77 //         searchQueries_map.put(2, new String[] { "Furniture" });
78 //         searchQueries_map.put(3, new String[] { "Gifts" });
79
80         for (Map.Entry<Integer, String[]> m : searchQueries_map.entrySet()) {
81             System.out.println(m.getKey() + " " + Arrays.toString(m.getValue()));
82         }
83     }
84
85     /*
86      * Main function that scraps amazon
87      */
88     public static void scrapAndSave() throws ParserConfigurationException,
       IOException, SAXException {
89         for (Map.Entry<Integer, String[]> topic : searchQueries_map.entrySet()) {
90             for (int topic_queries = 0; topic_queries < topic.getValue().length;
       topic_queries++) {
91                 for (int page = 1; page < 2; page++) {
92                     try {
93                         HtmlPage urlHTML = webClient.getPage(
94                                 AMAZON_PREFIX_URL + topic.getValue()[topic_queries
       ] + "&crid=2JOW4XXQM1KWM&sprefix="
95                                         + topic.getValue()[topic_queries] + "%2
       Caps%2C220&ref=sr_pg_" + page);
96                         webClient.getCurrentWindow().getJobManager().removeAllJobs
       ();
97
98                         List<HtmlElement> searchResults_List = urlHTML
99                                 .getByXPath("//div[@data-component-type='s-search-
       result']");
100                        int max = Math.min(searchResults_List.size(), 10);
101                        for (int searchResult = 0; searchResult < max;
       searchResult++) {
102                            HtmlDivision divv = (HtmlDivision) searchResults_List.
       get(searchResult);
103
104                            StringBuilder xmlStringBuilder = new StringBuilder();
105                            xmlStringBuilder.append("<?xml version=\"1.0\"?>");
106                            xmlStringBuilder.append(divv.asXml());
107
```

```
                            ByteArrayInputStream input = new ByteArrayInputStream(
                                    xmlStringBuilder.toString().getBytes(
    StandardCharsets.UTF_8));
                            xmlParser(input,
                                    DataBaseManager.LOCAL_IMG_FOLDER + '/' + Main.
    Topics[topic.getKey()].toLowerCase()
                                            + '/' + topic.getValue()[topic_queries
    ] + searchResult + ".webp",
                                    Main.Topics[topic.getKey()].toLowerCase());
                        }
                    } catch (IOException e) {
                        System.out.println("An error occurred: " + e);
                    }

                }
            }

        }
    }

    /*
     * Function that uses Xpath to go through the xml code, parse it and then
    return
     * the necessary strings.
     */
    public static void xmlParser(ByteArrayInputStream inputFile, String
    imageFilePath, String Topic)
            throws ParserConfigurationException, IOException, SAXException {

        String productName = "Sadly Not Found", productPrice = "Sadly Not Found",
    productImagePath = "Sadly Not Found";

        Document doc = builder.parse(inputFile);

        NodeList nListImages = doc.getElementsByTagName("img");
        Element imageElement = (Element) nListImages.item(0);
        String[] allImageURLs = imageElement.getAttribute("srcset").split(",");
        String hdImageIrl = allImageURLs[allImageURLs.length - 1].split(" ")[1];

        productName = imageElement.getAttribute("alt");
        productName = productName.replace(",", " -");
        if (productName.contains("Sponsored Ad - ")) {
            productName = productName.replace("Sponsored Ad - ", "");
        }
        System.out.println(productName);

        saveImage(hdImageIrl, imageFilePath);
        productImagePath = imageFilePath.replace(".webp", ".png");

        NodeList nList = doc.getElementsByTagName("span");
        for (int i = 0; i < nList.getLength(); i++) {
            Element currElement = (Element) nList.item(i);
            if (currElement.getAttribute("class").equals("a-price-whole")) {
                System.out.println("Price is: ");
                productPrice = currElement.getTextContent().replace(".", "");
                productPrice = productPrice.strip().replace(",", "");
                System.out.println(productPrice);
            }
        }
```

```
161         String[] data = new String[] { productName, productPrice, productImagePath
    };
162         if (productName.equalsIgnoreCase("Sadly Not Found") || productPrice.
    equalsIgnoreCase("Sadly Not Found")
163                 || productImagePath.equalsIgnoreCase("Sadly Not Found")) {
164             System.out.println("Not adding this data");
165         } else {
166             if (!Main.isLocalDatabaseUpToDate) {
167                 DataBaseManager.addDataToCSV(DataBaseManager.LOCAL_CSV_FOLDER + '/
    ' + Topic + ".csv", data);
168             }
169             if (Main.usingMongo) {
170                 if (!Main.isMongoUpToDate) {
171                     MongoManager.addDataToMongo(Topic, data);
172                 }
173             }
174         }
175     }
176 
177     /*
178      * Simple function save the image, but coz we cant work with webp images, we
179      * gotta convert them to png and save them right away.
180      */
181     public static void saveImage(String URLst, String filepath) {
182         if (new File(filepath).exists()) {
183             System.out.println("File Exists, gonna replace it");
184             new File(filepath).delete();
185         }
186 
187         try (InputStream in = new URL(URLst).openStream()) {
188             Files.copy(in, Paths.get(filepath));
189             converter = new Converter(filepath);
190             filepath = filepath.replace(".webp", ".png");
191             converter.convert(filepath, options);
192             Files.delete(Path.of(filepath.replace(".png", ".webp")));
193 
194         } catch (IOException e) {
195             System.out.println("we got some issue here with this file");
196         }
197     }
198 
199 }
```

Listing 13: Main Java file