ADDIS ABABA UNIVERISTY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

Center of Information Technology and Scientific Computing

Department of Software Engineering

Design Pattern

Assignment I – OOP Concepts

Prepared By:

Name – Mihret Tamene

Id –ATR/3534/08

Submitted To:

Mr Alazar

Date: Dec 30/2018

Addis Ababa, Ethiopia

Assignment 1 – OOP Concepts

1. **Write a program using any OOP language demonstrate command line argument processing.**

Command Line Argument is information passed to the program when you run the program. The passed information is stored as a string array in the main method.

Syntax: `>java ProgramName value1 value2`

```
class CmdLineArgs{
    public static void main(String args[]){
       for(int i =0 ; i < args.length; i++){
           System.out.print("Argument " + i );
           System.out.println(" is = " + args[i]);
         }
    }
}
```

Compiling the java application: This will create a CmdLineArgs.class file.

```
> javac CmdLineArgs.java
```

Running the Java class with the arguments

```
>java CmdLineArgs hi hello bye
```

OutPut

```
C:\Users\HP\Desktop\cmd Line>java CmdLineArgs hi hello bye
Argument 0 is = hi
Argument 1 is = hello
Argument 2 is = bye
```

```
>java CmdLineArgs one two three four five
```

```
C:\Users\HP\Desktop\cmd Line>java CmdLineArgs one two three four five
Argument 0 is = one
Argument 1 is = two
Argument 2 is = three
Argument 3 is = four
Argument 4 is = five
```

2. **Write a program using any OOP language demonstrate overloading.**

Method Overloading is a feature that allows a class to have more than one method having the same name, if their argument lists are different.

Three ways to overload a method

- Different Number of parameters
- Different Data type of parameters
- Different Sequence of Data type of parameters.

Number of parameters

```
class DisplayOverloding{
    public void display(String name){
        System.out.println("Display 1");
        System.out.println("My name is: " + name);
    }

    public void display(String name, int age){
        System.out.println("Display 2");
        System.out.println("My name is: " + name + "\nI am " + age + " Years old.");
    }
}


public class Main {

    public static void main(String[] args) {
        DisplayOverloding object = new DisplayOverloding();
        object.display("Abebe");
        object.display("Abebe", 20);
    }
}
```

Output

```
Display 1

My name is: Abebe

Display 2

My name is: Abebe

I am 20 Years old.

Process finished with exit code 0
```

Data type of parameters

```
class DisplayOverloding2{

    public void display(String name){
        System.out.println("Display 1");
        System.out.println("My name is: " + name);
    }

    public void display( int age){
        System.out.println("Display 2");
        System.out.println("Age is: " + age );
    }
}
public class Main {

    public static void main(String[] args) {
        DisplayOverloding2 object = new DisplayOverloding2();
        object.display("Abebe");
        object.display(20);
    }
}
```

Output

```
Display 1

My name is: Abebe

Display 2

Age is: 20

Process finished with exit code 0
```

Sequence of Data type of parameters

```
class DisplayOverloding3{

    public void display(String name, int age){
        System.out.println("Display 1");
        System.out.println("My name is: " + name);
    }

    public void display(int age, String name){
        System.out.println("Display 2");
        System.out.println("Age is: " + age + \n Name is: " + name );
    }
}

public class Main {

    public static void main(String[] args) {
       DisplayOverloding3 object = new DisplayOverloding3();
       object.display("Abebe", 20);
       object.display(20, "Abebe");
    }
}
```

Output

```
Display 1

My name is: Abebe

Display 2

Age is: 20

Name is: Abebe


Process finished with exit code 0
```

3. **Write a program using any OOP language demonstrate error handling using try, catch and finally blocks.**

   Exception is an event that interrupts the normal flow of execution. It is a disruption during the execution of the Java program.

- **Try block**: Normal code goes on this block.
- **Catch block**: If there is error in normal code, then it will go into this block
- **finally block:** executed irrespective of an exception being raised

Syntax:

```
try {
    //Statements that may cause an exception
}
catch {
   //Handling exception
}
finally {
   //Statements to be executed
}
```

Example

```
public class Main {

    public static void main(String[] args) {
        try {
            System.out.println("Inside try block");
            int num=10/0;
            int array[] = {1, 2 ,3};
            array[3] = 100;
        }
        catch (ArithmeticException e) {
            System.out.println("Catch clause due to Exception :  " + e);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Catch clause due to Exception : " + e);
        }
        finally{
            System.out.println("Inside the finally block");
        }
        System.out.println("End Of Main");
    }
}
```

Outputs

```
Inside try block

Catch clause due to Exception:  java.lang.ArithmeticException: / by zero

Inside the finally block

End Of Main

Process finished with exit code 0
```

Comment line which contains division by zero to get the Array Index out of Bound Exception.

```
Inside try block

Catch clause due to Exception : java.lang.ArrayIndexOutOfBoundsException: 3

Inside the finally block

End Of Main

Process finished with exit code 0
```

4. **Demonstrate used of virtual and override key words in OOP with simple program.**

   Declaring a method in **sub class** which is already present in **parent class** is known as method overriding. Overriding is done so that a child class can give its own implementation to a method which is already provided by the parent class.

   A Virtual function or virtual method is a function or method whose behavior can be overridden within an inheriting class by a function with the same signature to provide the polymorphic behavior.

```
class Person{
    private String name;
    protected char sex;
    public int age;

    Person(){
        name = null;
        sex = 'U';
        age = 0;
    }
    Person(String name, char sex, int age){
        this.name = name;
        this.sex =  sex;
        this.age = age;
    }
    String getName(){
        return name;
    }
    void Display(){
        System.out.println("name: " + name + " sex: " + sex + " age: " + age);
    }

}
```

```
class Student extends Person{
    private int rollNo;
    String branch;
    Student(String name, char sex, int age, int rollNo, String branch){
        super(name,sex,age);
        this.rollNo = rollNo;
        this.branch = branch;
    }
    @Override
    void Display(){
        System.out.println("name: " + getName()+ " sex: " + sex + " age: " + age + " rollNo: " +
rollNo + " branch: " + branch);
    }
}

public class Main {

    public static void main(String[] args) {
        Student s1 = new Student("Abebe", 'M', 20, 1, "computer science");
        Student  s2 = new Student("chaltu", 'F',  21, 2, "software enginnering");

        System.out.println("Student 1 details: ");
        s1.Display();

        Person p1 = new Person("kebede", 'M', 30);
        System.out.println("Person 1 details: ");

        p1.Display();
    }
}
```

Output

```
Student 1 details:

name: Abebe sex: M age: 20 rollNo: 1 branch: computer science

Person 1 details:

name: kebede sex: M age: 30



Process finished with exit code 0
```

5. **Write a program using any OOP language demonstrate delegates.**

Delegation is simply passing a duty off to someone/something else.

Delegation means that you use an object of another class as an instance variable, and forward messages to the instance.
Delegation can be viewed as a relationship between objects where one object forwards certain method calls to another object, called its delegate.

```
class RealPrinter {
    // the "delegate"
    void print() {
        System.out.println("The Delegate");
    }
}

class Printer {
    // the "delegator"
    RealPrinter p = new RealPrinter();

    // create the delegate
    void print(){
        p.print(); // delegation
    }
}
public class Main {

    public static void main(String[] args) {
        Printer printer = new Printer();
        printer.print();
    }
}
```

Output

```
The Delegate

Process finished with exit code 0
```

6. **Write a program using an OOP language demonstrate array of interface type (for running polymorphism).**

Polymorphism enables you to write programs that process objects that share the same superclass

Interface class

```
public interface Shapes {
    public void draw();
}
```

Concert classes

```
public class Circle implements Shapes {

    public void draw(){
        System.out.println("This is a Circle. ");
    }
}

public class Rectangle implements Shapes {
    @Override
    public void draw() {
        System.out.println("This is a Rectangle ");
    }
}

public class Triangle implements Shapes {
    @Override
    public void draw() {
        System.out.println("This is a Triangle. ");
    }
}
```

Main class

```
public class Main {

    public static void main(String[] args) {
        Shapes[] list = new Shapes[3];
       // create an array of Shape reference variables
        list[0] = new Circle();       // attach a Circle to first array slot
        list[1] = new Rectangle();    // attach a Rectangle to second slot
        list[2] = new Triangle();


        for (int i = 0; i < list.length; i++) {
            System.out.println("List " + i);
            list[0].draw();

        }
    }
}
```

7. Write a program using any OOP language to illustrate the use of different properties.

The properties object contains key and value pair both as a string. The java.util.Properties class is the subclass of Hash table. The Properties class provides methods to get data from the properties file and store data into the properties file. It is used to maintain lists of values in which the key is a String and the value is also a String.

String getProperty(String key) : Returns the value associated with the key. A null object is returned if the key is neither in the list nor in the default property list.

Set stringPropertyNames(): Returns a set of keys in this property list where the key and its corresponding value are strings

```
import java.util.Iterator;
import java.util.Properties;
import java.util.Set;

public class Main {

    public static void main(String[] args) {
        Properties courses = new Properties();
        Set name;
        String str;

        courses .put("DP", "Design Pattern");
        courses .put("DSP", "Distributed System Programming");
        courses .put("DBA", "Database Administration");
        courses .put("ITS", "IT Security ");
        courses .put("GIS", "Geographical information system");

        name = courses.keySet();    // get set-view of keys
        Iterator itr = name.iterator();

        while(itr.hasNext()) {
            str = (String) itr.next();
            System.out.println(str + " = " +
                    courses.getProperty(str) + ".");
        }

        Set set = courses.stringPropertyNames();
        System.out.print("\nProperty name in the set " + set);

    }
}
```

Output

```
DSP = Distributed System Programming.

DP = Design Pattern.

ITS = IT Security.

DBA = Database Administration.

GIS = Geographical information system.



Property name in the set [DP, DSP, ITS, DBA, GIS]
```