Experiment No.1

Date: 10-09-2020

Aim: To Solve Fundamentals of Java Programming

problems. CO Mapping – CO 1

Objective:

[1] To understand declaration of Classes, and Methods with its all features such as Constructors, Access Specifier
[2] Design and implement Inheritance, Polymorphism in JAVA
[3] Develop program to demonstrate Use of Static, final, super and this keyword, Garbage Collection

Concept:

Fundamentals of JAVA in this course is focusing on the characteristics of OOP. Java compiler makes between the primitive types and their corresponding object wrapper classes. The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. polymorphism in java by method overloading and method overriding. super and this keyword in Java. super keyword is used to access methods of the parent class while this is used to access methods of the current class. this is a reserved keyword in java i.e, we can't use it as an identifier. It represents the set of properties or methods that are common to all objects of one type. Inheritance represents the **IS-A relationship**, also known as *parent-child* relationship.

Lab Exercise: 1. Design a class CAR extending the sports car or luxury car. Show the flow of constructor in inheritance.

Code:

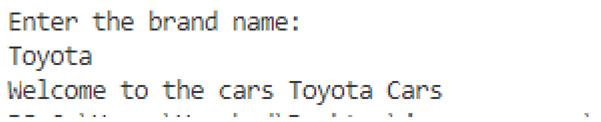
```
package Practical_first;

import java.util.Scanner;

class CAR{
    String var;
    CAR()
    {
        var = "";
    }
    Scanner sc = new Scanner(System.in);
    public void getdata()
    {
        System.out.println("Enter the brand name: ");
        var = sc.nextLine();
    }
    public void display()
    {
        System.out.println("Welcome to the cars " +var+ " Cars");
    }
}
public class sports extends CAR{

    sports()
    {
        super();
    }
    public static void main(String[] args)
    {
        sports c =new sports();
        c.getdata();
        c.display();
    }
}
```

Output:



```
Enter the brand name:
Toyota
Welcome to the cars Toyota Cars
```

Lab Exercise: 2. Suppose you have a Piggie Bank with an initial amount of $50 and you have to add some more amount to it. Create a class 'AddAmount' with a data member named 'amount' with an initial value of $50. Now make two constructors of this class as follows:
a. without any parameter - no amount will be added to the Piggie Bank
b. having a parameter which is the amount that will be added to Piggie Bank
c. Create object of the 'AddAmount' class and display the final amount in Piggie Bank.

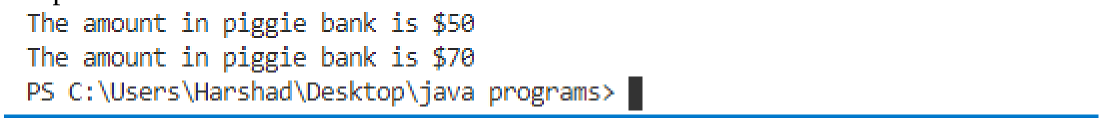
Code:

```
package Practical_first;

public class AddAmount {

    int amount,final_amount;
    AddAmount()
    {
        amount = 50;
    }
    AddAmount(AddAmount s1,int add_amount)
    {
        amount = s1.amount + add_amount;
    }
    void display()
    {
        System.out.println("The amount in piggie bank is $" +amount);
    }
    public static void main(String[] args)
    {
        AddAmount A = new AddAmount();
        AddAmount B = new AddAmount(A,20);
        A.display();
        B.display();
    }
}
```

Output:



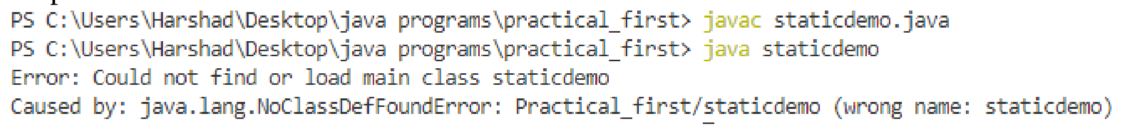
```
The amount in piggie bank is $50
The amount in piggie bank is $70
PS C:\Users\Harshad\Desktop\java programs>
```

Lab Exercise: 3. Initialized the data variables a and b of the staticdemo class using constructor and using static block. Don’t write main function. Show the output

Code:
```
package Practical_first;

class staticdemo
{
   int a,b,c;
   staticdemo()
   {
      a=10;
      b=5;
   }
   void add()
   {
      c=a+b;
      System.out.println("Additon = "+c);
   }
}
class demo{
   static
   {
      staticdemo s = new staticdemo();
      s.add();
   }
}
```

Output:



```
PS C:\Users\Harshad\Desktop\java programs\practical_first> javac staticdemo.java
PS C:\Users\Harshad\Desktop\java programs\practical_first> java staticdemo
Error: Could not find or load main class staticdemo
Caused by: java.lang.NoClassDefFoundError: Practical_first/staticdemo (wrong name: staticdemo)
```

Observation: Initially it seems very simple programs but later on problems on handling 3 rd experiment. I had used copy constructor in second program.

Conclusion: Have good hands on Construtors , inheritance and other oops concept.