OOP Lab: Experiment 4

Submitted By: Aryan Saxena
Batch: B1
SAP Id: 500082431
Roll No.: R214220274

Exercise 1: Write a Java program to show that private member of a super class cannot be accessed from derived classes.

Code:

LAB > Experiment 4 > Exp1.java > ...

```java
// Write a Java program to show that private member of a super class cannot be accessed from derived classes.
class Parent {
    int a;
    private int b;
    protected int c;

    Parent(int a, int b, int c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    public void print1() {
        System.out.println("Public method");
    }

    private void print2() {
        System.out.println("Private method");
    }

    protected void print3() {
        System.out.println("Protected method");
    }
}

class Child extends Parent {
    private int d;

    Child(int a, int b, int c, int d) {
        super(a, b, c);
        this.d = d;
    }
}

public class Exp1 {
    Run | Debug
    public static void main(String[] args) {
        Parent p = new Parent(1, 2, 3);
        p.print1();
        p.print3();
        //p.print2(); // Will show print2() has private access in Parent
        Child c = new Child(10, 20, 30, 40);
        c.print1();
        c.print3();
        //c.print2(); //Will show print2 is not visible to child class
    }
}
```

Output:

```
PS F:\UPES\Academics\2nd Year\3rd Semester\OOPs
Public method
Protected method
Public method
Protected method
PS F:\UPES\Academics\2nd Year\3rd Semester\OOPs
```

OUTPUT with Error- **The program takes care of repetition if digits are repeated**

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS F:\UPES\Academics\2nd Year\3rd Semester\OOPs Theory\sem3-Java-OOP> cd "f:\UPES\Academics\2nd Year\3rd Semester\OOPs Theory\sem3-Java-OOP\LAB\Experiment 4\" ; if ($?) { javac Exp1.java } ; if ($?) { java Exp1 }
Exp1.java:40: error: print2() has private access in Parent
        p.print2(); // Will show print2() has private access in Parent
         ^
Exp1.java:44: error: cannot find symbol
      c.print2(); //Will show print2 is not visible to child class
       ^
  symbol:   method print2()
  location: variable c of type Child
2 errors
PS F:\UPES\Academics\2nd Year\3rd Semester\OOPs Theory\sem3-Java-OOP\LAB\Experiment 4>
```

Exercise 2: Write a program in Java to create a Player class. Inherit the classes Cricket _Player, Football _Player and Hockey_ Player from Player class.

Code:

LAB > Experiment 4 > Exp2.java > Player

```java
// Write a program in Java to create a Player class. Inherit the classes Cricket _Player, Football _Player and Hockey_ Player from Player class.
class Player {
    protected String name;
    protected int age;
    protected double height;

    Player(String name, int age, double height) {
        this.name = name;
        this.age = age;
        this.height = height;
    }

    public void printDetails() {
        System.out.println("\nDetails of Player:");
        System.out.println("Name: "+name + "\n" +"Age: "+ age + "\n" + "Height: "+height);
    }
}

class Cricket_Player extends Player {
    private String teamName;
    private String position;

    Cricket_Player(String name, int age, double height, String teamName, String position) {
        super(name, age, height);
        this.teamName = teamName;
        this.position = position;
    }

    public void printDetails() {
        super.printDetails();
        System.out.println("Team Name: "+teamName + "\n" +"Position: "+ position);
    }
}

class Football_Player extends Player {
    private String teamName;
    private String position;

    Football_Player(String name, int age, double height, String teamName, String position) {
        super(name, age, height);
        this.teamName = teamName;
        this.position = position;
    }

    public void printDetails() {
        super.printDetails();
        System.out.println("Team Name: "+teamName + "\n" +"Position: "+ position);
    }
```

```java
    }

}

class Hockey_Player extends Player {
    private String teamName;
    private String position;

    Hockey_Player(String name, int age, double height, String teamName, String position) {
        super(name, age, height);
        this.teamName = teamName;
        this.position = position;
    }

    public void printDetails() {
        super.printDetails();
        System.out.println("Team Name: "+teamName + "\n" +"Position: "+ position);
    }

}
public class Exp2 {
    Run | Debug
    public static void main(String[] args) {
        Player p = new Cricket_Player("Amar", 30, 190, "Team1", "Batsman");
        p.printDetails();
        p = new Football_Player("Akbar", 25, 200.5, "Team2", "Extras");
        p.printDetails();
        p = new Hockey_Player("Anthony", 40, 170, "Team3", "Bowler");
    }
}
```

Output:

```
Details of Player:
Name: Amar
Age: 30
Height: 190.0
Team Name: Team1
Position: Batsman

Details of Player:
Name: Akbar
Age: 25
Height: 200.5
Team Name: Team2
Position: Extras
PS F:\UPES\Academics\2nd Year\3rd Semester\OOPs Theory\sem3-Java-OOP\LAB\Experiment 4>
```

Exercise 3: Write a class Worker and derive classes DailyWorker and SalariedWorker from it. Every worker has a name and a salary rate. Write method ComPay (int hours) to compute the week pay of every worker. A Daily Worker is paid on the basis of the number of days he/she works. The Salaried Worker gets paid the wage for 40 hours a week no matter what the actual hours are. Test this program to calculate the pay of workers. You are expected to use the concept of polymorphism to write this program.

Code:

```
 9  class Worker {
10
11      protected String name;
12      protected double salaryRate;
13
14      Worker(String name, double salaryRate) {
15          this.name = name;
16          this.salaryRate = salaryRate;
17      }
18  }
19  class SalariedWorker extends Worker {
20
21      private int hoursWorked;
22
23      SalariedWorker(String name, double salaryRate, int hoursWorked) {
24          super(name, salaryRate);
25          this.hoursWorked = hoursWorked;
26      }
27
28      public double ComPay() {
29          double salary = 40 * salaryRate;
30          return salary;
31      }
32  }
33  class DailyWorker extends Worker {
34
35      private int daysWorked;
36
37      DailyWorker(String name, double salaryRate, int daysWorked) {
38          super(name, salaryRate);
39          this.daysWorked = daysWorked;
40      }
41
42      public double ComPay() {
43          double salary = daysWorked * salaryRate;
44          return salary;
45      }
46  }
47  public class Exp3 {
        Run | Debug
48      public static void main(String[] args) {
49          SalariedWorker one = new SalariedWorker("Name1", 125, 37);
50          DailyWorker two = new DailyWorker("Name2", 100, 5);
51
52          System.out.println("Pay of Salaried Worker: "+one.ComPay());
53          System.out.println("Pay of Daily Worker: "+two.ComPay());
54      }
55  }
```

Output:

```
PS F:\UPES\Academics\2nd Year\3rd Semester\OOPs Theory\sem3-Java-OOP> cd "f:\UPES\Academics\2nd
Pay of Salaried Worker: 5000.0
Pay of Daily Worker: 500.0
PS F:\UPES\Academics\2nd Year\3rd Semester\OOPs Theory\sem3-Java-OOP\LAB\Experiment 4>
```

Exercise 4: Consider the trunk calls of a telephone exchange. A trunk call can be ordinary, urgent or lightning. The charges depend on the duration and the type of the call. Write a program using the concept of polymorphism in Java to calculate the charges.

Code:

```
4   class TrunkCall {
5       protected int duration;
6
7       TrunkCall(int duration) {
8           this.duration = duration;
9       }
10
11      public int getDuration() {
12          return this.duration;
13      }
14
15      public double cost() {
16          return 0.0;
17      }
18  }
19
20  class Ordinary extends TrunkCall {
21      private static final double charge = 1.0;
22
23      Ordinary(int duration) {
24          super(duration);
25      }
26
27      public double cost() {
28          return charge * duration;
29      }
30  }
31
32  class Urgent extends TrunkCall {
33      private static final double charge = 2.0;
34
35      Urgent(int duration) {
36          super(duration);
37      }
38
39      public double cost() {
40          return charge * duration;
41      }
42  }
43
44  class Lightning extends TrunkCall {
45      private static final double charge = 2.5;
46
47      Lightning(int duration) {
48          super(duration);
49      }
```

```java
    Lightning(int duration) {
        super(duration);
    }

    public double cost() {
        return charge * duration;
    }
}
public class Exp4 {
    Run | Debug
    public static void main(String[] args) {
        TrunkCall a = new Ordinary(10);
        TrunkCall b = new Urgent(10);
        TrunkCall c = new Lightning(10);
        System.out.println("Cost of Ordinary Call: "+a.cost());
        System.out.println("Cost of Urgent Call: "+b.cost());
        System.out.println("Cost of Lightning Call: "+c.cost());

    }
}

```

Output:

```
PS F:\UPES\Academics\2nd Year\3rd Semester\OOPs Theory\sem3-Java-OOP> cd "f:\UPES\Academi
Cost of Ordinary Call: 10.0
Cost of Urgent Call: 20.0
Cost of Lightning Call: 25.0
PS F:\UPES\Academics\2nd Year\3rd Semester\OOPs Theory\sem3-Java-OOP\LAB\Experiment 4>
```

Exercise 5: Design a class employee of an organization. An employee has a name, empid, and salary. Write the default constructor, a constructor with parameters (name, empid, and salary) and methods to return name and salary. Also write a method increaseSalary that raises the employee's salary by a certain user specified percentage. Derive a subclass Manager from employee. Add an instance variable named department to the manager class. Supply a test program that uses theses classes and methods.

Code:

```
 9  class Employee {
10      protected String name;
11      protected int empid;
12      protected double salary;
13
14      Employee(String name, int empid, double salary) {
15          this.name = name;
16          this.empid = empid;
17          this.salary = salary;
18      }
19
20      public double getSalary() {
21          return this.salary;
22      }
23
24      public String getName() {
25          return this.name;
26      }
27
28      public void increaseSalary(double percentage) {
29          this.salary += salary * percentage / 100;
30      }
31  }
32
33  class Manager extends Employee {
34      private String department;
35
36      Manager(String name, int empid, double salary, String department) {
37          super(name, empid, salary);
38          this.department = department;
39      }
40
41      public String getDepartment() {
42          return this.department;
43      }
44  }
45  public class Exp5 {
        Run | Debug
46      public static void main(String[] args) {
47          Employee A = new Employee("Bunty", 8118, 1024.67);
48          Manager B = new Manager("Bubly", 1881, 9525.25, "Department 1");
49          System.out.println("Name: "+A.getName()+"\nSalary: "+A.getSalary());
50          System.out.println("Name: "+B.getName()+"\nSalary: "+B.getSalary());
51      }
52  }
```

Output:

```
PS F:\UPES\Academics\2nd Year\3rd Semester\OOPs Theory\sem3-Java-OOP> cd "f:\UPES\Academi
Name: Bunty
Salary: 1024.67
Name: Bubly
Salary: 9525.25
PS F:\UPES\Academics\2nd Year\3rd Semester\OOPs Theory\sem3-Java-OOP\LAB\Experiment 4>
```