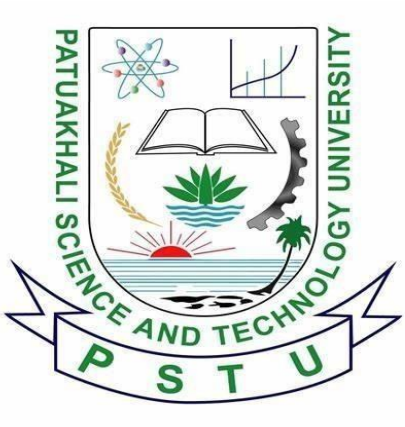
PATUAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY



Course Code: CCE-121

SUBMITTED TO:

Prof. Samsujjaman Sobuj

Department of Computer and Communication Engineering

Faculty of Computer Science And Engineering

SUBMITTED BY:

Name: MD Noushad Bhuiyan

ID: 2102038, Registration No: 10165

Faculty of Computer Science and Engineering

Date of submission: 12-15-2023

Java Chapter 8

8.1 Fill in the gaps:

a. Static import on demand
b. format
c. shadows.
d. public services, public interface.
e. single-type-import.
f. default constructor.
g. toString.
h. accessor methods, query methods.
i. predicate.
j. values.
k. has-a.
l. enum.
m. static.
n. single static import.
o. principle of least privilege.
p. final.
q. type-import-on-demand.
r. mutator methods.
s. BigDecimal.
t. throw

8.2 Package uses and negative sides:

Use:

In Java, package access, also known as package-private or default access, is a level of access control that restricts the visibility of classes, interfaces, and members (fields and methods) to only other classes and interfaces within the same package.

Nagitive aspect:

When a class or member has package access, it means that it is not accessible outside its package, even if it is declared as public. The absence of an access modifier (i.e., no modifier specified) implies package access by default.

8.3 State an example where you can reuse the constructor of a parent class in Java.

Animal class

```
class Animal{
    String name;
    public Animal(String name){
        this.name = name;
    }

    public void isEating(){
        System.out.println(name + " is eating");
    }

}
```

Dog class

```
class dog extends Animal{
    String breed;

    public dog(String name,String breed){
        super(name);
        this.breed=breed;
    }
    public void isBreeding(){
        System.out.println(name + " is eating " +"than "+ breed);
    }
}
```

Main class

```
public class test{
   public static void main(String[] args){
    dog d = new dog("dog", "labrador");
        d.isEating();
        d.isBreeding();
   }

}
```

8.4 Cylinder Class

Cylinder Class

```
public class Cylinder{

    private double height;
    private double width;

    void setHeight(double height){
        this.height=height;
    }
    double getHeight()
    {
        return this.height;
    }
    void setWidth(double width){
        this.width=width;
    }
    double getWidth()
    {
        return this.width;
    }
    void Area()
    {
        double area=3.14159*height*width;
        System.out.println("Area of Cylinder is "+area);
    }

}
```

Main Class

```
public class Main{
    public static void main(String[] args){
        Cylinder c = new Cylinder();
        c.setHeight(10);
        c.setWidth(10);
        c.Area();
    }
}
```

8.6 Savings Account Class

Saving_Account Class

```
public class Savings_Account{
    static double annualInterestRate;
    private double savingsBalance;
    void setSavingsBalance(double savingsBalance) {
        this.savingsBalance = savingsBalance;
    }
    double getSavingsBalance() {
        return savingsBalance;
    }
    double lateMonthlyInterest()
    {
        return (savingsBalance * annualInterestRate)/12;
    }
     void modifyInterestRate( double annualInterestRate)
    {
        this.annualInterestRate = annualInterestRate;
    }
}
```

Main test class

```
public class main{public static void main(String[] args) {
    Savings_Account saver1= new Savings_Account();
    Savings_Account saver2= new Savings_Account();
    saver1.setSavingsBalance(2000);
    saver2.setSavingsBalance(3000);
    Savings_Account.annualInterestRate=4;
    double saver1MonthlyInterest=saver1.lateMonthlyInterest();
    double saver2MonthlyInterest=saver2.lateMonthlyInterest();
    System.out.println("when interest is "+ saver1.annualInterestRate+"% than
monthly interest for saver 1 is "+ saver1MonthlyInterest);
    System.out.println("when interest is "+ saver2.annualInterestRate+"% than
monthly interest for saver 2 is "+ saver2MonthlyInterest);
    saver1.modifyInterestRate(5.0);
    saver2.modifyInterestRate(5.0);
    saver1MonthlyInterest=saver1.lateMonthlyInterest();
    saver2MonthlyInterest=saver2.lateMonthlyInterest();
    System.out.println("when interest is "+ saver1.annualInterestRate+"% than
monthly interest for saver 1 is "+ saver1MonthlyInterest);
    System.out.println("when interest is "+ saver2.annualInterestRate+"% than
monthly interest for saver 2 is "+ saver2MonthlyInterest);}}
```

8.9 Generate Random Number

```java
import java.security.SecureRandom;
import java.util.Scanner;

public class GenerateRandomNumber{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the value of n: ");
        int n = scanner.nextInt();

        generateRandomNumbers(n);

        scanner.close();
    }

    private static void generateRandomNumbers(int n) {
        SecureRandom random = new SecureRandom();

        System.out.println("Generated Random Numbers:");
        for (int i = 0; i < n; i++) {
            int randomNumber = random.nextInt(91) + 10;
            System.out.println(randomNumber);
        }
    }
}
```

8.10 Use of ENUM

```
public class FoodEnum{
    enum Food {
        APPLE("Fruit", 95),
        BANANA("Fruit", 105),
        CARROT("Vegetable", 25);

        private final String type;
        private final int calories;

        Food(String type, int calories) {
            this.type = type;
            this.calories = calories;
        }

        public String getType() {
            return type;
        }

        public int getCalories() {
            return calories;
        }
    }

    public static void main(String[] args) {
        System.out.println("Food Information:");
        for (Food food : Food.values()) {
            System.out.println("Name: " + food.name());
            System.out.println("Type: " + food.getType());
            System.out.println("Calories: " + food.getCalories() + " calories");
            System.out.println();
        }
    }
}
```

8.11 Complex Number

```
public class Complex {
    private double realPart;
    private double imaginaryPart;

    public Complex() {
        this.realPart = 0.0;
        this.imaginaryPart = 0.0;
    }

    public Complex(double realPart, double imaginaryPart) {
        this.realPart = realPart;
        this.imaginaryPart = imaginaryPart;
    }
      public Complex add(Complex other) {
        double newRealPart = this.realPart + other.realPart;
        double newImaginaryPart = this.imaginaryPart + other.imaginaryPart;
        return new Complex(newRealPart, newImaginaryPart);
    }
      public Complex subtract(Complex other) {
        double newRealPart = this.realPart - other.realPart;
        double newImaginaryPart = this.imaginaryPart - other.imaginaryPart;
        return new Complex(newRealPart, newImaginaryPart);
    }
      public void print() {
        System.out.println("(" + this.realPart + ", " + this.imaginaryPart +
")");
    }
      public static void main(String[] args) {
       Complex complex1 = new Complex(2.5, 3.0);
        Complex complex2 = new Complex(1.5, 2.0);
      Complex sumResult = complex1.add(complex2);
        System.out.print("Sum: ");
        sumResult.print();
      Complex subtractResult = complex1.subtract(complex2);
        System.out.print("Difference: ");
        subtractResult.print();
    }
}
```