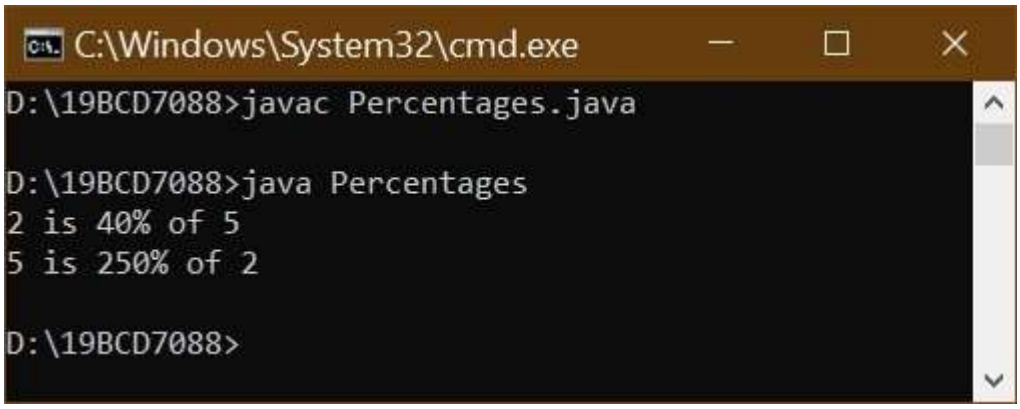
1.a.

Create an application named Percentages whose main() method holds two double variables. Assign values to the variables. Pass both variables to a method named computePercent() that displays the two values and the value of the first number as a percentage of the second one. For example, if the numbers are 2.0 and 5.0, the method should display a statement similar to “2.0 is 40 percent of 5.0.” Then call the method a second time, passing thevalues in reverse order. Save the application as Percentages.java.

```
public class Percentages{
		static void computePercent(int a, int b){
		System.out.println(a+" is "+(a*100/b)+"% of "+b);
	}
	public static void main(String[] args) {
		int a = 2;
		int b = 5;
		computePercent(2,5);
		computePercent(5,2);
	}

}
```

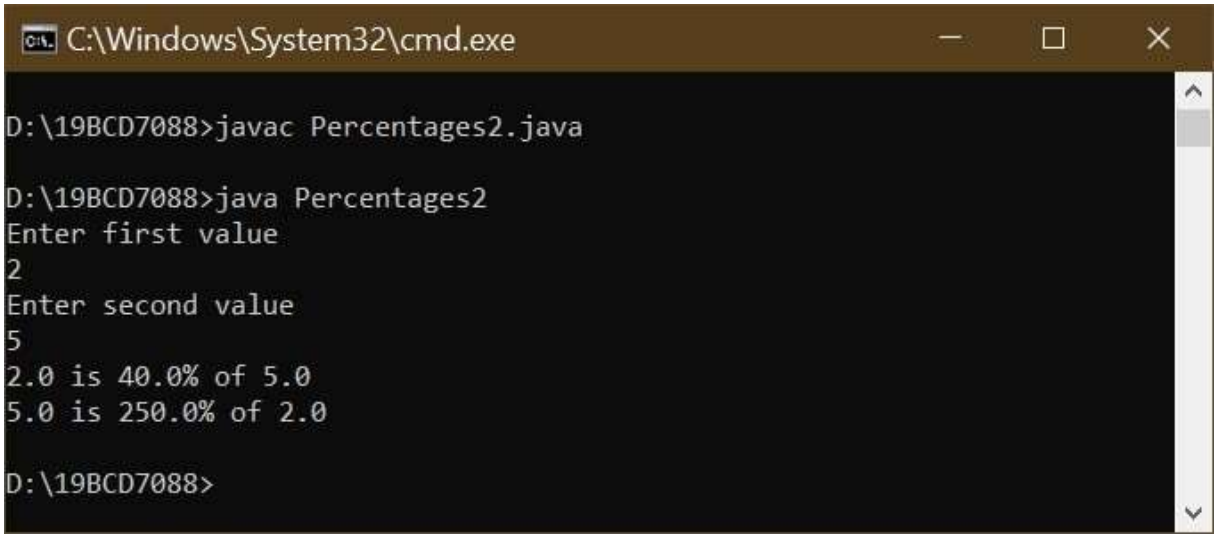


1.b.

Modify the Percentages class to accept the values of the two doubles from auser at the keyboard. Save the file as Percentages2.java.

```
import java.util.*;


public class Percentages2{
```

```
	static void computePercent(double a, double b){
		System.out.println(a+" is "+(a*100/b)+"% of "+b);
	}
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter first value");
		double a = sc.nextDouble();
		System.out.println("Enter second value");
		double b = sc.nextDouble();
		computePercent(a,b);
		computePercent(b,a);
	}

}
```

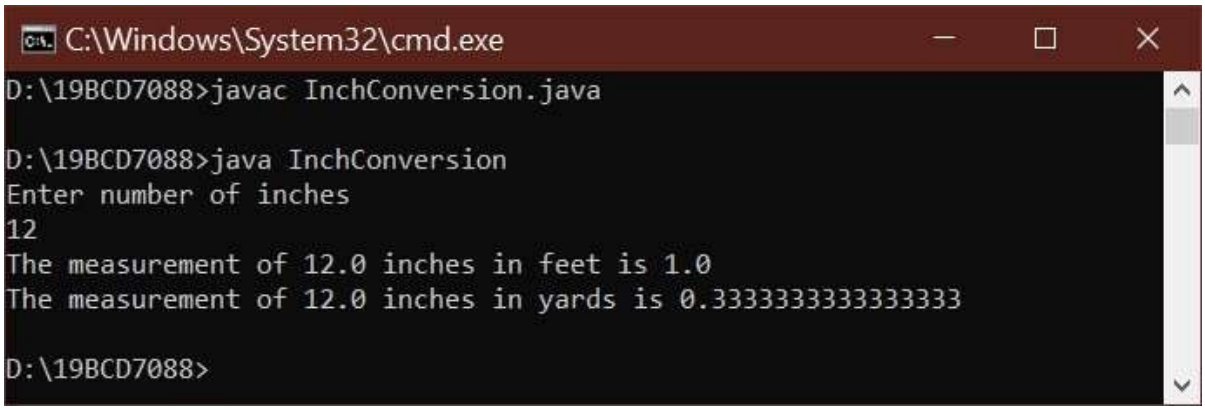


2.

There are 12 inches in a foot and 3 feet in a yard. Create a class named InchConversion. Its main() method accepts a value in inches from a user at the keyboard, and in turn passes the entered value to two methods. One converts the value from inches to feet, and the other converts the same value from inches to yards. Each method displays the results with appropriate explanation. Save the application as InchConversion.java.

```
import java.util.*;

public class InchConversion{
```

```
	static void InchtoFeet(double a){
		System.out.println("The measurement of "+ a +" inches in feet is "+ a/12);
	}
	static void InchtoYards(double b){
		System.out.println("The measurement of "+ b +" inches in yards is "+ b/36);
	}
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter number of inches");
		double a = sc.nextDouble();
		InchtoFeet(a);
		InchtoYards(a);
	}
}
```



3.

Assume that a gallon of paint covers about 350 square feet of wall space. Create an application with a main() method that prompts the user for the length, width, and height of a rectangular room. Pass these three values to a method that does the following:

•• Calculates the wall area for a room

•• Passes the calculated wall area to another method that calculates and returns the number of gallons of paint needed

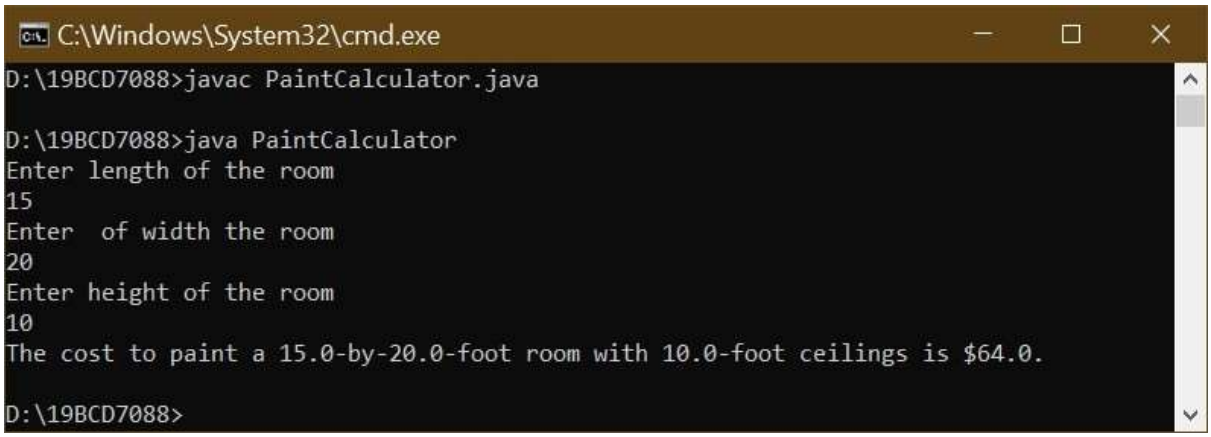
•• Displays the number of gallons needed

•• Computes the price based on a paint price of $32 per gallon, assuming that the painter can buy any fraction of a gallon of paint at the same price as a whole gallon

•• Returns the price to the main() method

The main() method displays the final price. For example, the cost to paint a 15-by-20-foot room with 10-foot ceilings is $64. Save the application as PaintCalculator.java.

```
import java.util.*;


public class PaintCalculator{
	static double area (double l,double b,double h){
		return ((2*l*h)+(2*b*h));
	}
	static double gallons(double q){
		return q/350;
	}
	static double price(double g){
		return g*32;
	}
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter length of the room");
		double l = sc.nextDouble();
		System.out.println("Enter of width the room");
		double b = sc.nextDouble();
		System.out.println("Enter height of the room");
		double h = sc.nextDouble();
		double a = area(l,b,h);
		double gallon = gallons(a);
		double rate = price(gallon);
		System.out.println("The cost to paint a " + l + "-by-" + b + "-foot room with " + h + "-
foot ceilings is $" + rate + ".");
	}
}
```

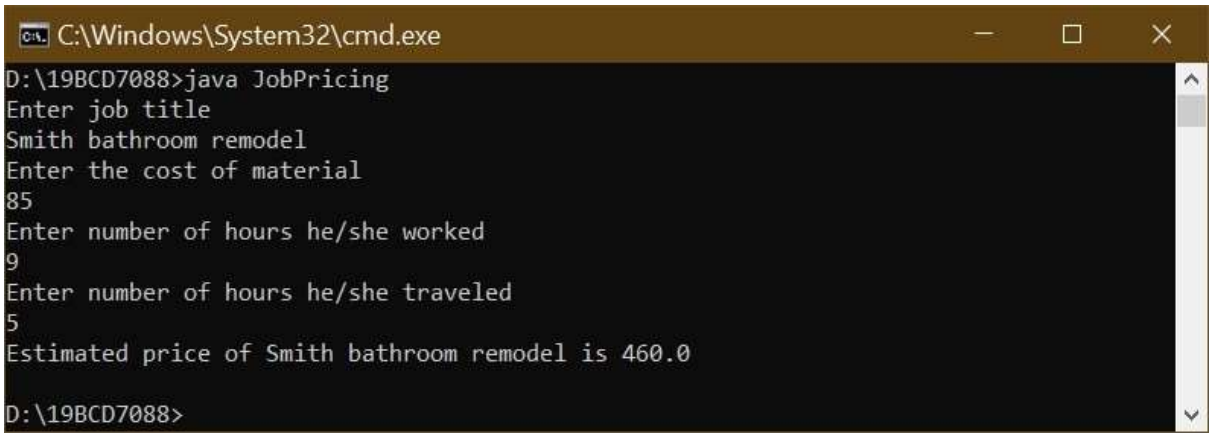


4.

Herbert's Home Repair estimates each job cost as the cost of materials plus $35 per hour while on the job, plus $12 per hour for travel time to the job site. Create a class that contains a main() method that prompts the user for the name of a job (for example, Smith bathroom remodel), the cost of materials, the number of hours of work required, and the number of hours travel time. Pass the numeric data to a method that computes estimate for the job and returns the computed value to the main() method where the job name and estimated price are

displayed. Save the program as JobPricing.java.

```
import java.util.*;
public class JobPricing{
	static double computetion (double m,double h,double t){
		return m+(h*35)+(t*12);
	}
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);

		System.out.println("Enter job title");

		String str = sc.nextLine();

		System.out.println("Enter the cost of material");

		double m  = sc.nextDouble();

		System.out.println("Enter number of hours he/she worked");

		double h  = sc.nextDouble();

		System.out.println("Enter number of hours he/she traveled");

		double t  = sc.nextDouble();

		double n = computetion(m,h,t);

		System.out.println("Estimated price of " + str + " is " + n);
	}
}
```

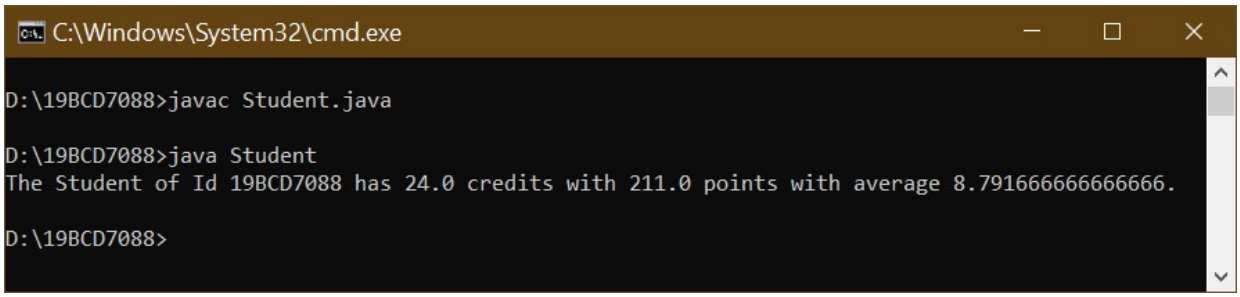


5.a.

Create a class named Student that has fields for an ID number, number of credit hours earned, and number of points earned. (For example, many schools compute grade point averages based on a scale of 4, so a three-credit-hour class in which a student earns an A is worth 12 points.) Include methods to assign values to all fields. A Student also has a field for grade point average. Include a method to compute the grade point average field by dividing points by credit hours earned. Write methods to display the values in each Student field. Save this class as Student.java.

```
public class Student{
        static String str;
        static double
         credit,points; static void
         setId(String s){
                str=s;
        }
        static void setCredit(int c){
                credit=c;
        }
        static void setPoints(int p){
                points=p;
```

```
	}
	static void average(){
		double a = points/credit;
		System.out.println("The Student of Id " + str + " has " + credit + " credits with " +
points + " points with average " + a + ".");
	}
	public static void main(String[] args) {
		setId("19BCD7088");
		setCredit(24);
		setPoints(211);
		average();
	}
}
```

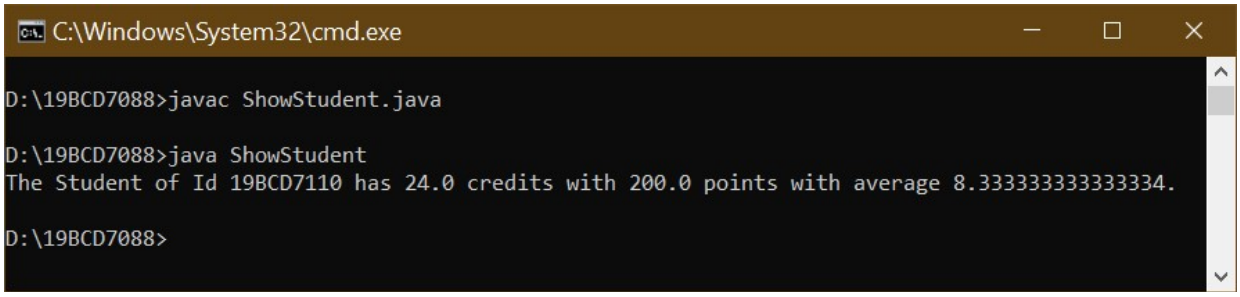


5.b.

Write a class named ShowStudent that instantiates a Student object from the class you created and assign values to its fields. Compute the Student grade point average, and then display all the values associated with the Student. Save the application as ShowStudent.java.

```
class Student{
	String str;
	double
	 credit,points;

	 void setId(String s){
		str=s;
	}
```

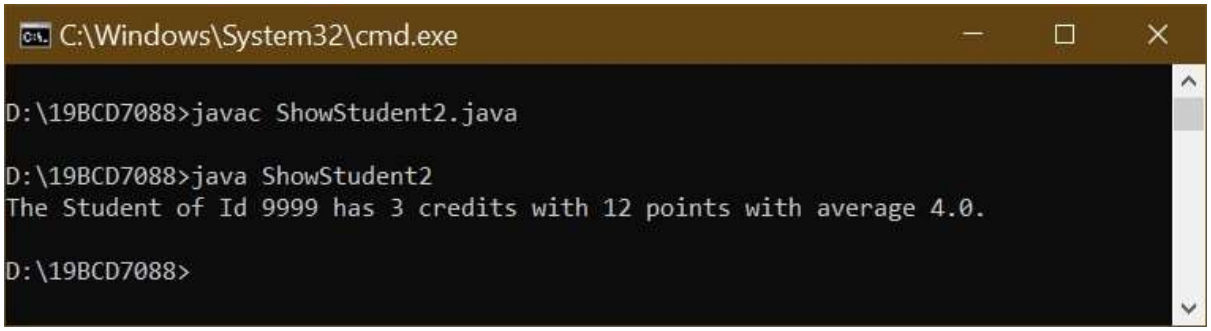
```
	void setCredit(int c){
		credit=c;
	}
	void setPoints(int p){
		points=p;
	}
	void average(){
		double a = points/credit;
		System.out.println("The Student of Id " + str + " has " + credit + " credits with " +
points + " points with average " + a + ".");
	}
}

public class ShowStudent{
	public static void main(String[] args) {
		Student st = new Student();
		st.setId("19BCD7110");
		st.setCredit(24);
		st.setPoints(200);
		st.average();
	}
}
```



5.c.

Create a constructor for the Student class you created. The constructor should initialize each Student's ID number to 9999, his or her points earned to 12, and credit hours to 3 (resulting in a grade point average of 4.0). Write a program that demonstrates that the constructor works by instantiating an object and displaying the initial values. Save the application as ShowStudent2.java.

```
class Student{
        String str;
        double credit,points;
        Student(String s,int c,int p){
                str=s;
                credit=c;
                points=p;
        }
        void mean(){
                double a = points/credit;
                System.out.println("The Student of Id " + str + " has " + credit + " credits with " +
points + " points with average " + a + ".");
        }
}
public class ShowStudent2{
        public static void main(String[] args) {
        Student st = new Student("9999",3,12);
        st.mean();
}


}
```

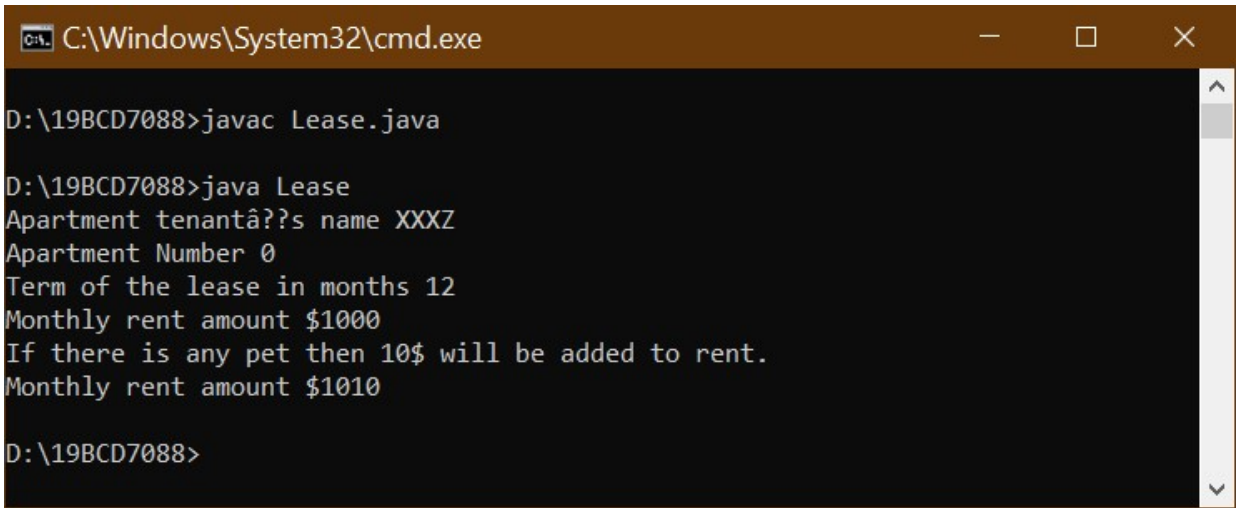


6.a.

Create a class named Lease with fields that hold an apartment tenant's name, apartment number, monthly rent amount, and term of the lease in months. Include a constructor that initializes the name to "XXX", the apartment number to 0, the rent to 1000, and the term to 12. Also include methods to get and set each of the fields. Include a nonstatic method named addPetFee() that adds $10 to the monthly rent value and calls a static method named explainPetPolicy() that explains the pet fee. Save the class as Lease.java.

```
class Data{
	String str;
	int a,r,m;
	Data(String str,int a ,int r, int m){
		this.str=str;
		this.a=a;
		this.r=r;
		this.m=m;
	}
	void setName(String s){
		this.str=s;
	}
	void setAptNum(int a){
		this.a=a;
	}
	void setRent(int r){
		this.r=r;
	}
	void setMonth(int m){
		this.m=m;
	}
	void getName(){
		System.out.println("Apartment tenant's name " + this.str);
	}
	void getAptNum(){
```

```
		System.out.println("Apartment Number " + this.a);
	}
	void getMonth(){
		System.out.println("Term of the lease in months " + this.m);
	}
	void getRent(){
		System.out.println("Monthly rent amount $" + this.r);
	}
	void addPetFee(){
		this.r= this.r+10;
	}
	static void explainPetPolicy(){
		System.out.println("If there is any pet then 10$ will be added to rent.");
	}

}
public class Lease{
	public static void main(String[] args) {
		Data d = new Data("XXXZ",0,1000,12);
		d.getName();
		d.getAptNum();
		d.getMonth();
		d.getRent();
		d.explainPetPolicy();
		d.addPetFee();
		d.getRent();


	}
}
```



6.b.

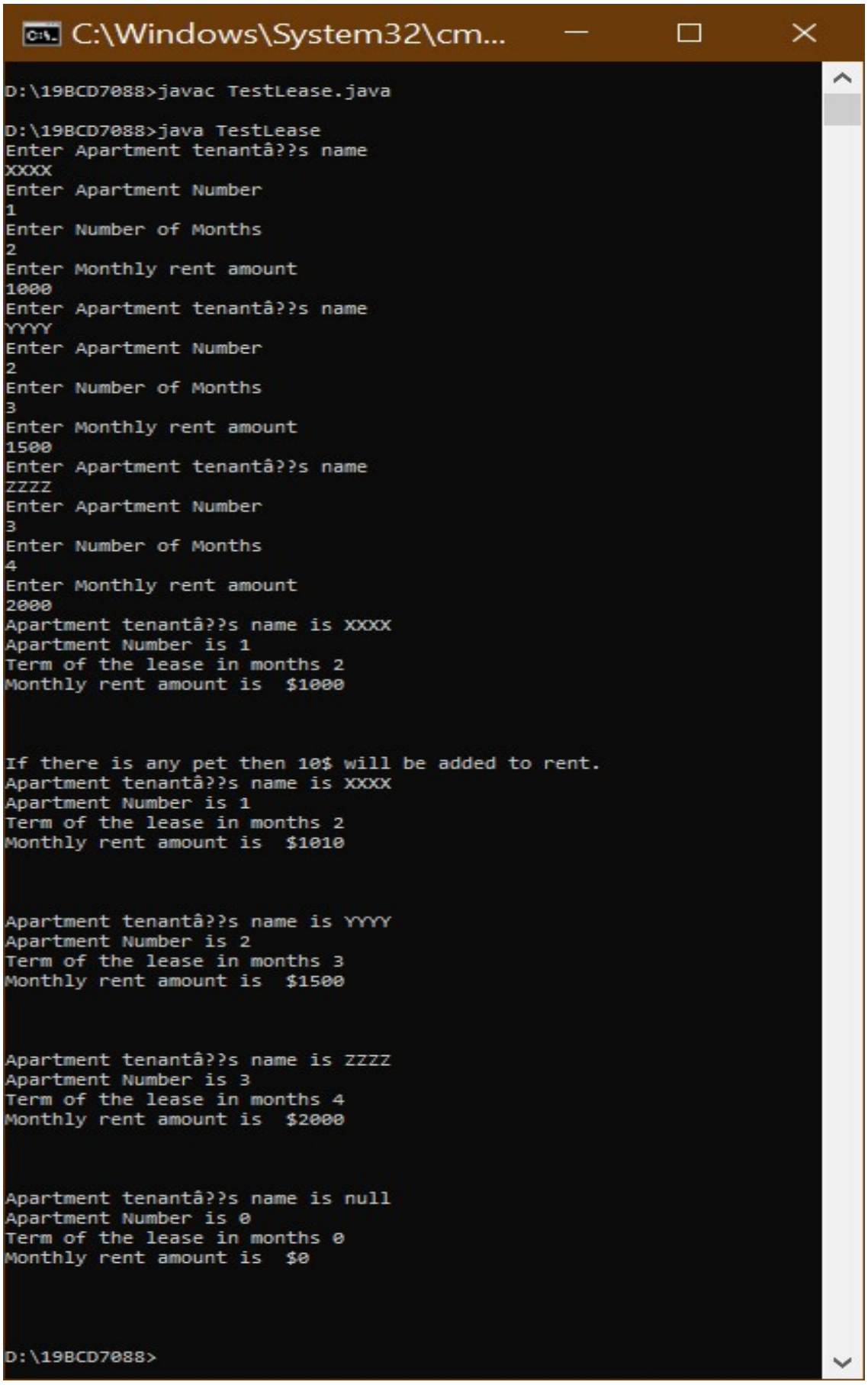
Create a class named TestLease whose main() method declares four Lease objects. Call a getData() method three times. Within the method, prompt a user for values for each field for a Lease, and return a Lease object to the main() method where it is assigned to one of main()'s Lease objects. Do not prompt the user for values for the fourth Lease object, but let it continue to hold the default values. Then, in main(), pass one of the Lease objects to a showValues() method that displays the data. Then call the addPetFee() method using the passed Lease object and confirm that the fee explanation statement is displayed. Next, call the showValues() method for the Lease object again and confirm that the pet fee has been added to the rent. Finally, call the showValues() method with each of the other three objects; confirm that two hold the values you supplied as input and one holds the constructor default values. Save the application as TestLease.java.

```
import java.util.*;
class Lease{
	String str;
	int a,r,m;
	void setName(String s){
		this.str=s;
	}
	void setAptNum(int a){
		this.a=a;
	}
	void setRent(int r){
		this.r=r;
	}
	void setMonth(int m){
```

```
        this.m=m;
}
void getName(){
        System.out.println("Apartment tenant's name is " + this.str);
}
void getAptNum(){
        System.out.println("Apartment Number is " + this.a);
}
void getMonth(){
        System.out.println("Term of the lease in months " + this.m);
}
void getRent(){
        System.out.println("Monthly rent amount is $" + this.r);
}
void addPetFee(){
        this.r= this.r+10;
}
Lease getData(){
        Scanner sc = new Scanner(System.in);
        Lease temp = new Lease();
        System.out.println("Enter Apartment tenant's name");
        String e = sc.nextLine();
        System.out.println("Enter Apartment Number");
        int k =sc.nextInt();
        System.out.println("Enter Number of Months");
        int o = sc.nextInt();
        System.out.println("Enter Monthly rent amount");
        int l = sc.nextInt();
        temp.str=e;
        temp.a=k;
        temp.r=l;
```

```
                temp.m=o;
                return temp;

        }
        void showValues(){
                getName();
                getAptNum();
                getMonth();
                getRent();
                System.out.println("\n\n");
        }
        static void explainPetPolicy(){
                System.out.println("If there is any pet then 10$ will be added to rent.");
        }
}
public class TestLease{
        public static void main(String[] args) {
                Lease l1 = new Lease();
                Lease l2 = new Lease();
                Lease l3 = new Lease();
                Lease l4 = new Lease();
                l1=l1.getData();
                l2=l2.getData();
                l3=l3.getData();
                l1.showValues();
                l1.addPetFee();
                l1.explainPetPolicy();
                l1.showValues();
                l2.showValues();
                l3.showValues();
                l4.showValues();
```

```
    }
}
```


C:\Windows\System32\cm...
D:\19BCD7088>javac TestLease.java
D:\19BCD7088>java TestLease
Enter Apartment tenantâ??s name
XXXX
Enter Apartment Number
1
Enter Number of Months
2
Enter Monthly rent amount
1000
Enter Apartment tenantâ??s name
YYYY
Enter Apartment Number
2
Enter Number of Months
3
Enter Monthly rent amount
1500
Enter Apartment tenantâ??s name
ZZZZ
Enter Apartment Number
3
Enter Number of Months
4
Enter Monthly rent amount
2000
Apartment tenantâ??s name is XXXX
Apartment Number is 1
Term of the lease in months 2
Monthly rent amount is $1000
If there is any pet then 10$ will be added to rent.
Apartment tenantâ??s name is XXXX
Apartment Number is 1
Term of the lease in months 2
Monthly rent amount is $1010
Apartment tenantâ??s name is YYYY
Apartment Number is 2
Term of the lease in months 3
Monthly rent amount is $1500
Apartment tenantâ??s name is ZZZZ
Apartment Number is 3
Term of the lease in months 4
Monthly rent amount is $2000
Apartment tenantâ??s name is null
Apartment Number is 0
Term of the lease in months 0
Monthly rent amount is $0
D:\19BCD7088>

1.

Create a class named Billing that includes three overloaded computeBill() methods for a photo book store.

•• When computeBill() receives a single parameter, it represents the price of one photo book ordered. Add 8% tax, and return the total due.

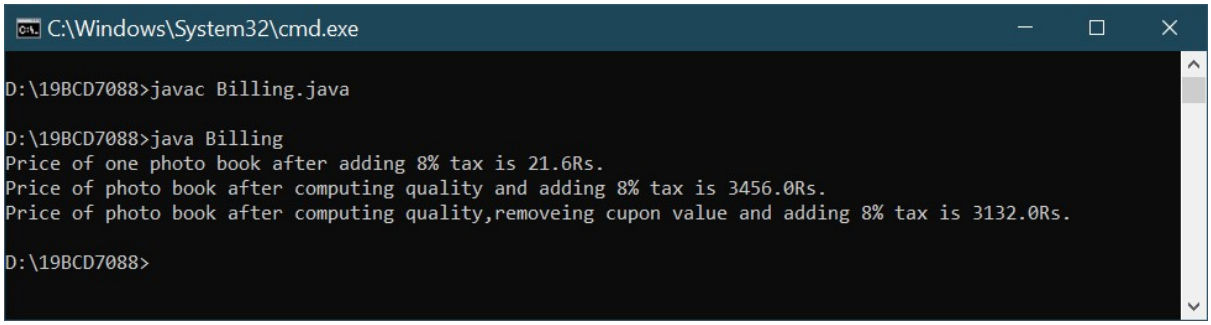
•• When computeBill() receives two parameters, they represent the price of a photo book and the quantity ordered. Multiply the two values, add 8% tax, and return the total due.

•• When computeBill() receives three parameters, they represent the price of a photo book, the quantity ordered, and a coupon value. Multiply the quantity and price, reduce the result by the coupon value, and then add 8% tax and return the total due.

Write a main() method that tests all three overloaded methods. Save the application as Billing.java.

```
public class Billing{
	static double computeBill(double a){
		return a*1.08;
	}
	static double computeBill(double a,double b){
		return(a*b)*1.08;
	}
	static double computeBill(double a, double b,double c){
		double n = (a*b)-c;
		return n*1.08;
	}
	public static void main(String[] args) {
		double k = computeBill(20);
		double l = computeBill(40,80);
		double m = computeBill(50,60,100);
		System.out.println("Price of one photo book after adding 8% tax is " + k + "Rs.");
		System.out.println("Price of photo book after computing quality and adding 8% tax is " + l + "Rs.");
```

```
        System.out.println("Price of photo book after computing quality,removeing cupon value and adding 8% tax is " + m + "Rs.");
    }

}
```



2.

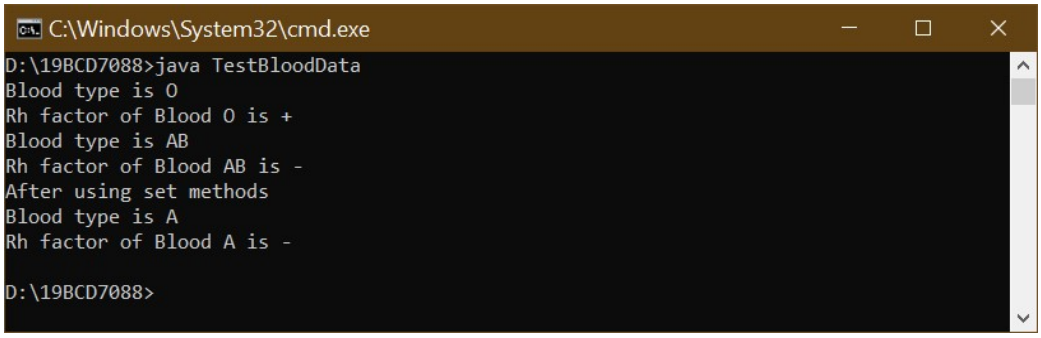
a.

Create a class named BloodData that includes fields that hold a blood type (the four blood types are O, A, B, and AB) and an Rh factor (the factors are 1 and –). Create a default constructor that sets the fields to O and 1, and an overloaded constructor that requires values for both fields. Include get and set methods for each field. Save this file as BloodData.java. Create an application named TestBloodData that demonstrates each method works correctly. Save the application as TestBloodData.java.

```
class BloodData{
    String Blood,Rh;
    BloodData(){
        this.Blood = "O";
        this.Rh = "+";
    }
    BloodData(String Blood,String Rh){
        this.Blood=Blood;
        this.Rh=Rh;
    }
    void setBlood(String Blood){
        this.Blood=Blood;
```

```
	}
	void setRh(String Rh){
		this.Rh=Rh;
	}
	void getBlood(){
		System.out.println("Blood type is " + this.Blood);
	}
	void getRh(){
		System.out.println("Rh factor of Blood " + this.Blood + " is " + this.Rh);
	}
}
```

```
public class TestBloodData{
	public static void main(String[] args) {
	BloodData b1 = new BloodData();
	BloodData b2 = new BloodData("AB","-");
	b1.getBlood();
	b1.getRh();
	b2.getBlood();
	b2.getRh();
	b1.setBlood("A");
	b1.setRh("-");
	System.out.println("After using set methods");
	b1.getBlood();
	b1.getRh();
}
}
```



2.

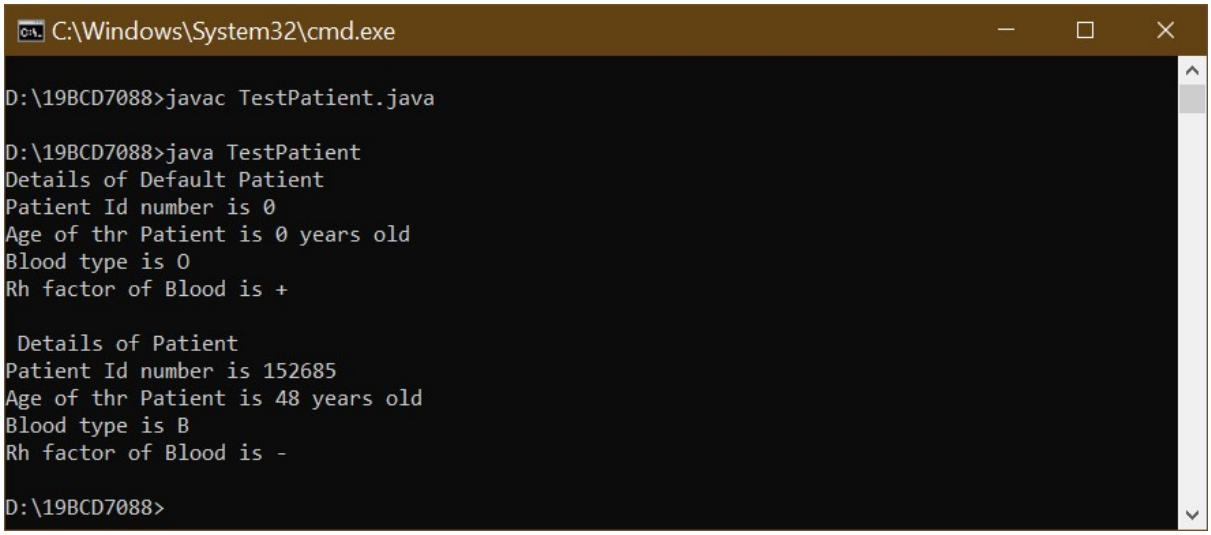
b.

Create a class named Patient that includes an ID number, age, and BloodData. Provide a default constructor that sets the ID number to 0, the age to 0, and the BloodData values to O and 1. Create an overloaded constructor that provides values for each field. Also provide get methods for each field. Save the file as Patient.java. Create an application that demonstrates hat each method works correctly, and save it as TestPatient.java.

```
class Patient{
        int Id,age;
        String Blood,Rh;
        Patient(){
                this.Id = 0;
                this.age =0;
                this.Rh="+";
                this.Blood = "O";
        }
        Patient(int Id,int age,String Blood,String Rh){
                this.Id = Id;
                this.age=age;
                this.Rh=Rh;
                this.Blood = Blood;
        }
        void getId(){
                System.out.println("Patient Id number is " + this.Id);
        }
```

```
	void getage(){
		System.out.println("Age of thr Patient is " + this.age + " years old");
	}
	void getBlood(){
		System.out.println("Blood type is " + this.Blood);
	}
	void getRh(){
		System.out.println("Rh factor of Blood is " + this.Rh);
	}
}
```

```
public class TestPatient{
	public static void main(String[] args) {
		Patient p1 = new Patient();
		Patient p2 = new Patient(152685,48,"B","-");
		System.out.println("Details of Default Patient");
		p1.getId();
		p1.getage();
		p1.getBlood();
		p1.getRh();
		System.out.println("\n Details of Patient");
		p2.getId();
		p2.getage();
		p2.getBlood();
		p2.getRh();
	}
}
```



3.

a.

Create a class named Circle with fields named radius, diameter, and area. Include a constructor that sets the radius to 1 and calculates the other two values. Also include methods named setRadius() and getRadius(). The setRadius() method not only sets the radius, but it also calculates the other two values. (The diameter of a circle is twice the radius, and the area of a circle is pi multiplied by the square of the radius. Use the Math class PI constant for this calculation.) Save the class as Circle.java.

```
import java.lang.Math;
class Circle{
        double r;
        double d;
        double area;
        Circle(){
                this.r = 1;
                this.d = 2*1;
                this.area = Math.PI*1*1;
        }
        void setRadius(double r){
                this.r = r;
                this.d = 2*r;
                this.area = Math.PI*r*r;
```

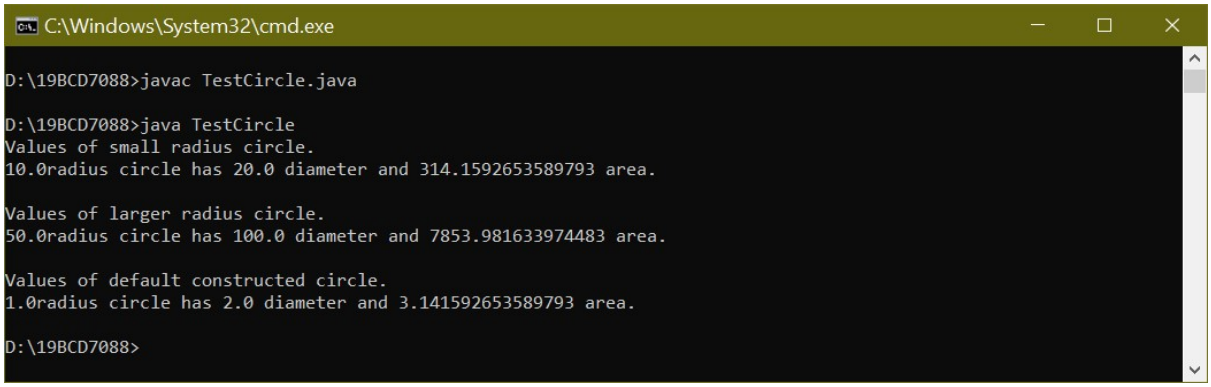
```
	}
	void getRadius(){
		System.out.println(this.r + "radius circle has " + this.d + " diameter and " + this.area
+ " area.");
	}
}
```

3.

b.

Create a class named TestCircle whose main() method declares several Circle objects. Using the setRadius() method, assign one Circle a small radius value, and assign another a larger radius value. Do not assign a value to the radius of the third circle; instead, retain the value assigned at construction. Display all the values for all the Circle objects. Save the application as TestCircle.java.

```
public class TestCircle{
	public static void main(String[] args) {
		Circle c1 = new Circle();
		Circle c2 = new Circle();
		Circle c3 = new Circle();
		c1.setRadius(10);
		c2.setRadius(50);
		System.out.println("Values of small radius circle.");
		c1.getRadius();
		System.out.println("\nValues of larger radius circle.");
		c2.getRadius();
		System.out.println("\nValues of default constructed circle.");
		c3.getRadius();
	}
```



4.

Write a Java application that uses the Math class to determine the answers for each of the following:(Use java.lang.Math class)

a. The square root of 37

b. The sine and cosine of 300

c. The value of the floor, ceiling, and round of 22.8

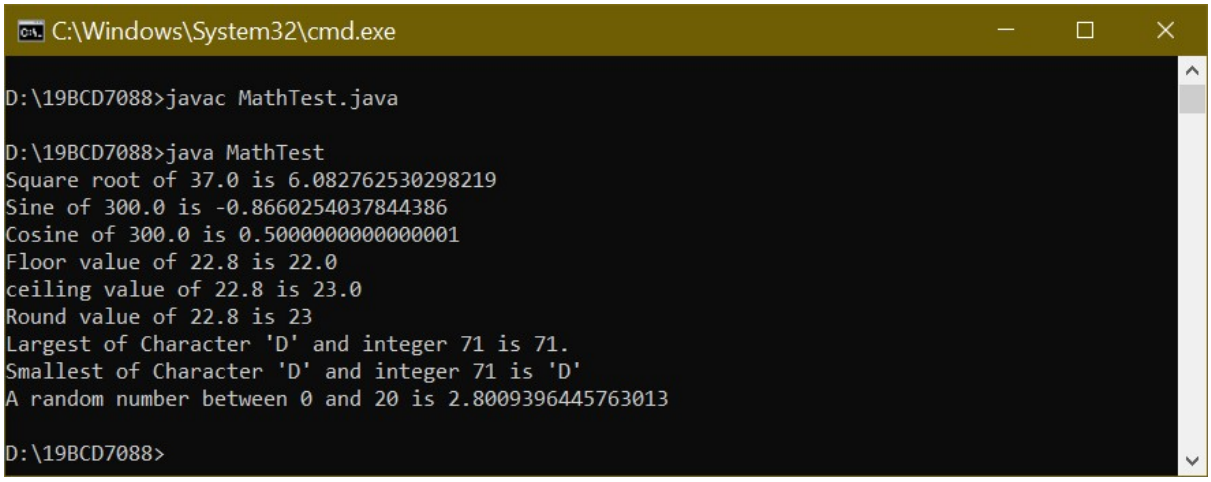
d. The larger and the smaller of the character ‘D’ and the integer 71

e. A random number between 0 and 20 (Hint: The random() method returns a value between 0 and 1; you want a number that is 20 times larger.)

Save the application as MathTest.java.

```
import java.lang.Math;
public class MathTest{
	public static void main(String[] args) {
		double a = 37;
		System.out.println("Square root of " + a + " is " + Math.sqrt(a));
		double b = 300;
		double n=Math.toRadians(b);
		System.out.println("Sine of " + b + " is " + Math.sin(n));
		System.out.println("Cosine of " + b + " is " + Math.cos(n));
		double c = 22.8;
		System.out.println("Floor value of " + c + " is " + Math.floor(c));
		System.out.println("ceiling value of " + c + " is " + Math.ceil(c));
		System.out.println("Round value of " + c + " is " + Math.round(c));
		if ((char)(Math.max('D',71)) == 'D'){
			System.out.println("Largest of Character 'D' and integer 71 is 'D'");
		}
		else {
			System.out.println("Largest of Character 'D' and integer 71 is 71.");
		}
```

```
        if ((char)(Math.min('D',71)) == 'D'){
            System.out.println("Smallest of Character 'D' and integer 71 is 'D'");
        }
        else {
            System.out.println("Smallest of Character 'D' and integer 71 is 71.");
        }
        System.out.println("A random number between 0 and 20 is " + 20*Math.random());

    }
}
```



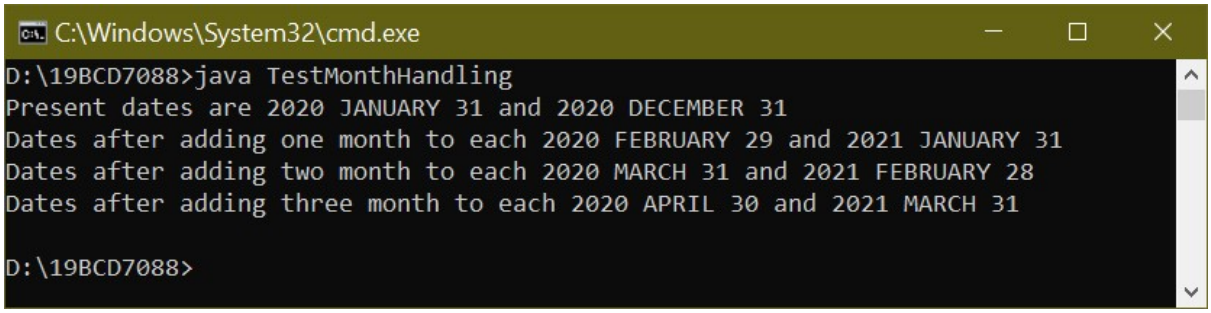
5.

a.

Write a program that declares two LocalDate objects and assign values that represent January 31 and December 31 in the current year. Display output that demonstrates the dates displayed when one, two, and three months are added to each of the objects. Save the application as TestMonthHandling.java.

```
import java.time.*;

public class TestMonthHandling{
    public static void main(String[] args) {
        LocalDate mon1;
        LocalDate mon2;
        LocalDate temp1;
```

```
        LocalDate temp2;

        int mo1 = 1;

        int day1 = 31;

        int year = 2020;

        mon1 = LocalDate.of(year, mo1, day1);

        int mo2 = 12;

        int day2 = 31;

        mon2 = LocalDate.of(year, mo2, day2);

        System.out.println("Present dates are " +  mon1.getYear()  +" " + mon1.getMonth()+
" " + mon1.getDayOfMonth() + " and " +  mon2.getYear()  +" " + mon2.getMonth()+" " +
mon2.getDayOfMonth());

        temp1 = mon1.plusMonths(1);

        temp2 = mon2.plusMonths(1);

        System.out.println("Dates after adding one month to each " + temp1.getYear() +" " +
temp1.getMonth() + " " + temp1.getDayOfMonth() + " and " +  temp2.getYear() +" " +
temp2.getMonth()+" " + temp2.getDayOfMonth() );

        temp1 = mon1.plusMonths(2);

        temp2 = mon2.plusMonths(2);

        System.out.println("Dates after adding two month to each " + temp1.getYear() + " "+
temp1.getMonth() + " " + temp1.getDayOfMonth() + " and " +  temp2.getYear() +" " +
temp2.getMonth()+" " + temp2.getDayOfMonth() );

        temp1 = mon1.plusMonths(3);

        temp2 = mon2.plusMonths(3);

        System.out.println("Dates after adding three month to each " + temp1.getYear() + "
" +  temp1.getMonth() + " " + temp1.getDayOfMonth() + " and " +  temp2.getYear() +" " +
temp2.getMonth()+" " + temp2.getDayOfMonth() );

    }

}
```

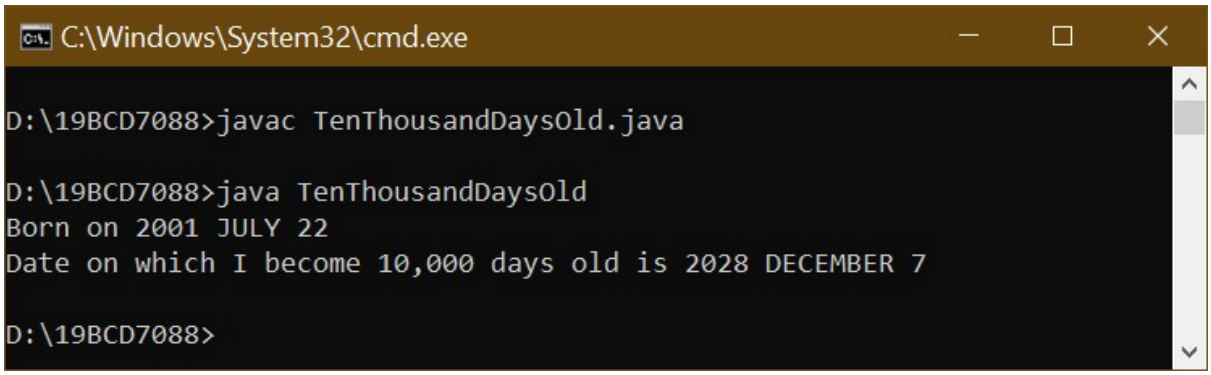


5.

b.

Write an application that computes and displays the day on which you become (or became) 10,000 days old. Save the application as TenThousandDaysOld.java.

```
import java.time.*;

public class TenThousandDaysOld{

	public static void main(String[] args) {

		LocalDate dob;

		int mo = 7;

		int day = 22;

		int year = 2001;

		dob = LocalDate.of(year, mo, day);

		System.out.println("Born on " +  dob.getYear()  +" " + dob.getMonth() + " " + dob.getDayOfMonth());

		dob=dob.plusDays(10000);

		System.out.println("Date on which I become 10,000 days old is " + dob.getYear()  +" " + dob.getMonth()+ " " + dob.getDayOfMonth());

	}

}
```



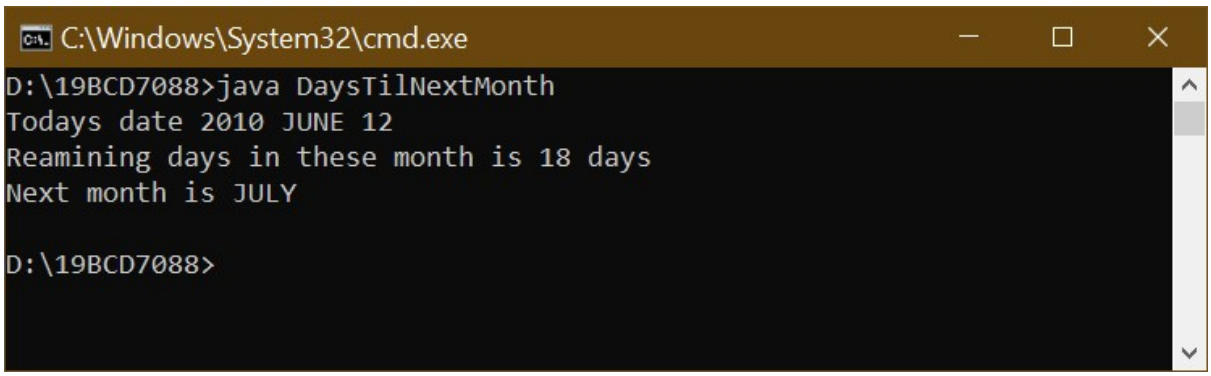
5.

c.

The LocalDate class includes an instance method named lengthOfMonth() that returns the number of days in the month. Write an application that uses methods in the LocalDate class to calculate how

many days are left until the first day of next month. Display the result, including the name of the next month. Save the file as DaysTilNextMonth.java.

```
import java.time.*;

public class DaysTilNextMonth{

	public static void main(String[] args) {

		LocalDate mon;

		LocalDate temp;

		int mo = 6;

		int day = 12;

		int year = 2010;

		mon = LocalDate.of(year, mo, day);

		int remain = mon.lengthOfMonth() - day;

		System.out.println("Todays date " +  mon.getYear() + " " + mon.getMonth()+ " " +
mon.getDayOfMonth());

		System.out.println("Reamining days in these month is " + remain + " days");

		mon = mon.plusMonths(1);

		System.out.println("Next month is " +  mon.getMonth());


	}

}
```



Lab3

L1 slot

V.Manikanta Bhuvanesh

19BCD7088

1.

Mick's Wicks makes candles in various sizes. Create a class for the business named Candle that contains data fields for color, height, and price. Create get methods for all three fields. Create set methods for color and height, but not for price. Instead, when height is set, determine the price as $2 per inch. Create a child class named ScentedCandle that contains an additional data field named scent and methods to get and set it. In the child class, override the parent's setHeight() method to set the price of a ScentedCandle object at $3 per inch. Write an application that instantiates an object of each type and displays the details. Save the files as Candle.java, ScentedCandle.java, and DemoCandles.java.

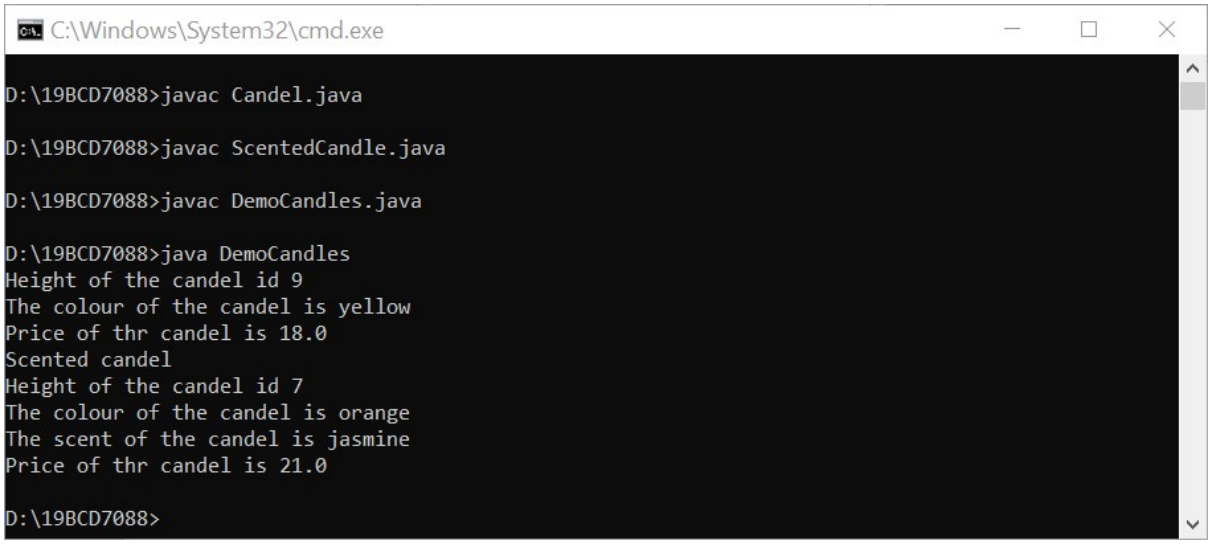
```
class Candle
{
String color;
int height;
double price;
void getColor(){
System.out.println ("The colour of the candel is " + color);
}
void getHeight()
{
System.out.println("Height of the candel id " + height);
}
void getPrice()
{
System.out.println("Price of thr candel is " + price);
}
void setColor(String c)
{
color = c;
}
void setHeight(int h)
{
height=h;
```

```
double n = 2;
price = h * n;
}
}
```

```
class ScentedCandle extends Candle
{
 String scent;
 void getScent()
{
System.out.println("The scent of the candel is " + scent);
}
void setScent(String s)
{
scent = s;
}
void setHeight(int h)
{
double n = 3;
super.setHeight(h);
price = h * n;
}
}
```

```
public class DemoCandles
{
public static void main(String args[])
{
Candle a = new Candle();
ScentedCandle b = new ScentedCandle();
a.setColor("yellow");
a.setHeight(9);
b.setColor("orange");
```

```
b.setScent("jasmine");
b.setHeight(7);
a.getHeight();
a.getColor();
a.getPrice();
System.out.println("Scented candel");
b.getHeight();
b.getColor();
b.getScent();
b.getPrice();
}
}
```



2.

Create a class named Poem that contains fields for the name of the poem and the number of lines in it. Include a constructor that requires values for both fields. Also include get methods to retrieve field values. Create three subclasses: Couplet, Limerick, and Haiku. The constructor for each subclass requires only a title; the lines field is set using a constant value. A couplet has two lines, a limerick has five lines, and a haiku has three lines. Create an application that demonstrates usage of an object of each type. Save the files as Poem.java, Couplet.java, Limerick.java, Haiku.java, and DemoPoems.java.

```
class Poem
{
String name;
int lines;

Poem(String name, int lines)
{
this.name = name;
this.lines = lines;
}

String getPoemName()
{
return name;
}

int getLines()
{
return lines ;
}
}
```

```
class Couplet extends Poem
{
Couplet(String name)
```

```
{
super(name,2);
}
}
```

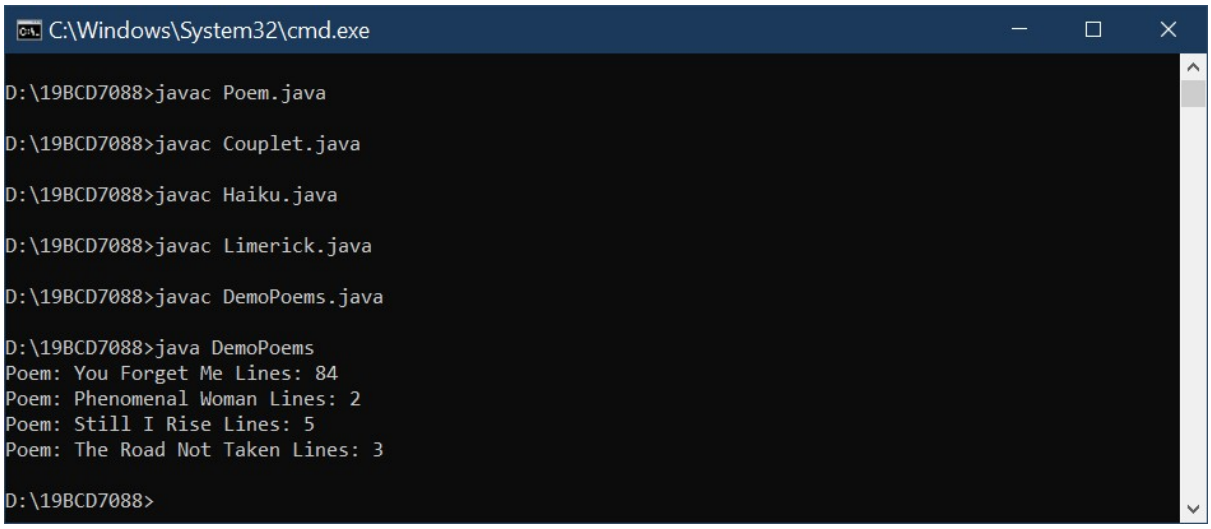
```
class Haiku extends Poem
{
Haiku(String name)
{
super(name,3);
}
}
```

```
class Limerick extends Poem
{
Limerick(String name)
{
super(name,5);
}
}
```

```
public class DemoPoems
{
public static void main(String[] args)
{
Poem p1 = new Poem("You Forget Me", 84);
Couplet p2 = new Couplet("Phenomenal Woman");
Limerick p3 = new Limerick("Still I Rise");
Haiku p4 = new Haiku("The Road Not Taken");
System.out.println("Poem: " + p1.getPoemName() +" Lines: " + p1.getLines());
System.out.println("Poem: " + p2.getPoemName() +" Lines: " + p2.getLines());
System.out.println("Poem: " + p3.getPoemName() +" Lines: " + p3.getLines());
System.out.println("Poem: " + p4.getPoemName() +" Lines: " + p4.getLines());
```

}

}



3

The developers of a free online game named "Sugar Smash" have asked you to develop a class named SugarSmashPlayer that holds data about a single player. The class contains the following fields: the player's integer ID number, a String screen name, and an array of integers that stores the highest score achieved in each of 10 game levels. Include get and set methods for each field. The get and set methods for the scores should each require two parameters—one that represents the score achieved and one that represents the game level to be retrieved or assigned. Display an error message if the user attempts to assign or retrieve a score from a level that is out of range for the array of scores. Additionally, no level except the first one should be set unless the user has earned at least 100 points at each previous level. If a user tries to set a score for a level that is not yet available, issue an error message. Create a class named PremiumSugarSmashPlayer that descends from SugarSmashPlayer. This class is instantiated when a user pays $2.99 to have access to 40 additional levels of play. As in the free version of the game, a user cannot set a score for a level unless the user has earned at least 100 points at all previous levels. Create a program that instantiates several objects of each type and demonstrates the methods. Save the files as SugarSmashPlayer.java, PremiumSugarSmashPlayer.java, and DemoSugarSmash.java.

```
class SugarSmashPlayer
{
  int ID;
  String screenName;
   protected int[] scores = new int[10];
  void setId(int num)
  {
    ID = num;
  }
```

```
void setName(String player)
{
  screenName = player;
}
void setScore(int score, int level)
{

  if (level == 0)
   { scores[level] = score;}
  else
  {
    if (scores[level - 1] >= 100 && level < scores.length)
      {scores[level] = score;}
    else
    {
      System.out.println("Invalid score");
    }
  }
}
int getId()
{
  return ID;
}
String getName(){
  return screenName;
}
int getScore(int level)
{
  if (level >= scores.length)
  {
    System.out.println("Invalid game level");
```

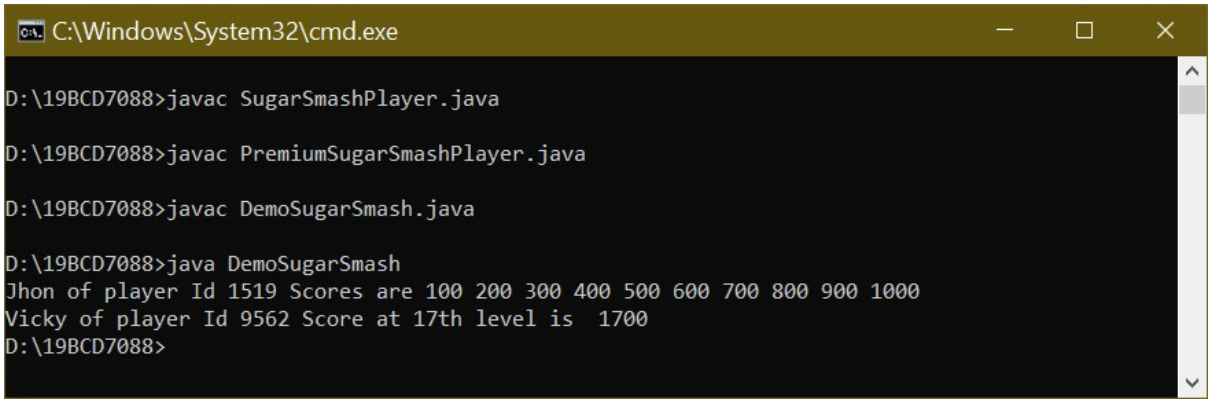
```
      return 0;
    }
    else{
      return scores[level];

    }

  }
}
```

```
class PremiumSugarSmashPlayer extends SugarSmashPlayer
{
  protected int[] scores = new int[50];
  void setScore(int score, int level)
  {

    if (level == 0)
     { scores[level] = score;}
    else
    {
      if (scores[level - 1] >= 100 && level < scores.length)
        {scores[level] = score;}
      else
      {
        System.out.println("Invalid score");
      }
    }
  }
  int getScore(int level)
  {
    if (level >= scores.length)
    {
```

```
      System.out.println("Invalid game level");
      return -1;
    }
    else{
      return scores[level];

    }

  }
}
```

```
public class DemoSugarSmash
{
  public static void main(String[] args)
  {
    SugarSmashPlayer ss = new SugarSmashPlayer();
    PremiumSugarSmashPlayer ps = new PremiumSugarSmashPlayer();
    ss.setName("Jhon");
    ss.setId(1519);
    int a= 100;
    for(int i = 0;i<10;i++){
        ss.setScore(a,i);
        a=a+100;
    }
    String name = ss.getName();
    int n = ss.getId();
    System.out.print(name + " of player Id " + n + " Scores are ");
    int o;
    for(int j=0;j<10;j++){
       o=ss.getScore(j);
    System.out.print(o + " ");
```

```
    }
    System.out.println();
    ps.setId(9562);
    ps.setName("Vicky");
    int b= 100;
    for(int k = 0;k<50;k++){
        ps.setScore(b,k);
        b=b+100;
    }
    name = ps.getName();
    n=ps.getId();
    System.out.print(name + " of player Id " + n + " Score at 17th level is  ");
    System.out.print(ps.getScore(16));


  }
}
```



4.

Create a class named CollegeCourse that includes data fields that hold the department (for example, ENG), the course number (for example, 101), the credits (for example, 3), and the fee for the course (for example, $360). All of the fields are required as arguments to the constructor, except for the fee, which is calculated at $120 per credit hour. Include a display() method that displays the course data. Create a subclass named LabCourse that adds $50 to the course fee. Override the parent class display() method to indicate that the course is a lab course and to display all the data. Write an application named UseCourse that prompts the user for course information. If the user enters a class in any of the following departments, create a LabCourse: BIO, CHM, CIS, or PHY. If the user enters any other department, create a CollegeCourse that does not include the lab fee. Then display the course data. Save the files as CollegeCourse.java, LabCourse.java, and UseCourse.java.

```
class CollegeCourse
{
 double Ch = 120.00;
 String dpt;
 int num;
 int c;
 double cf;
 CollegeCourse(String d, int n, int nc)
 {
    dpt = d.toUpperCase();
    num = n;
    c = nc;
    cf = Ch * c;
 }
 String getdpt()
 {
    return dpt;
 }
 int getcourseNo()

 {
    return num;
 }
```

```
 int getCredits()
 {
    return c;
 }
 double getcourseFee()
 {
    return cf;
 }
 void display()
 {
    System.out.println("department " + this.getdpt());
    System.out.println("Course number " + this.getcourseNo());
    System.out.println("Credit hours " + this.getCredits());
    System.out.println("Course fee " + this.getcourseFee());
 }

}
```

```
class LabCourse extends CollegeCourse

{
 double lf = 50.00;
 double cf;
 LabCourse(String dpt, int cn, int c)
 {
    super(dpt, cn, c);
    cf = super.getcourseFee() + lf;
 }
 double getLabCourseFee()
 {
    return cf;
```

```
 }
 void display()
 {
    System.out.println("department " + super.getdpt());
    System.out.println("Course number " + super.getcourseNo());
    System.out.println("Credit hours " + super.getCredits());
    System.out.println("Course fee " + this.getLabCourseFee());
     }

}
```

```
import java.util.Scanner;
public class UseCourse{
 public static void main(String[] args)
 {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter the department of the course: ");
    String dept = sc.nextLine();
    System.out.print("Enter the number of the courses ");
    int number = sc.nextInt();
    System.out.print("Enter the credit hours of the courses ");
    int hours = sc.nextInt();
    if(dept.equals("BIO") || dept.equals("CHM")|| dept.equals("CIS") || dept.equals("PHY"))
    {
       LabCourse l = new LabCourse(dept, number, hours);
       l.display();
    }
    else
    {
       CollegeCourse c = new CollegeCourse(dept, number, hours);
       c.display();
    }
```

```
  }


}
```


C:\Windows\System32\cmd.exe
D:\19BCD7088>javac LabCourse.java
D:\19BCD7088>javac UseCourse.java
D:\19BCD7088>java UseCourse
Enter the department of the course: BIO
Enter the number of the courses 9
Enter the credit hours of the courses 84
department BIO
Course number 9
Credit hours 84
Course fee 10130.0
D:\19BCD7088>

5.

Develop a set of classes for a college to use in various student service andpersonnel applications. Classes you need to design include the following:

•• Person—A Person contains a first name, last name, street address, zip code,and phone number. The class also includes a method that sets each datafield, using a series of dialog boxes and a display method that displays all of a Person's information on a single line at the command line on the screen.

•• CollegeEmployee—CollegeEmployee descends from Person. A CollegeEmployee also includes a Social Security number, an annual salary, and a department name, as well as methods that override the Person methods to accept and display all CollegeEmployee data.

•• Faculty—Faculty descends from CollegeEmployee. This class also includes Boolean field that indicates whether the Faculty member is tenured, as well as methods that override the CollegeEmployee methods to accept and display this additional piece of information.

•• Student—Student descends from Person. In addition to the fields available in Person, a Student contains a major field of study and a grade point average as well as methods that override the Person methods to accept and display these additional facts.

Write an application named CollegeList that declares an array of four "regular" CollegeEmployees, three Faculty, and seven Students. Prompt the user to specify which type of person's data will be entered (C, F, or S), or allow the user to quit (Q). While the user chooses to continue (that is, does not quit), accept data entry for the appropriate type of Person. If the user attempts to enter data for more than four CollegeEmployees, three Faculty, or seven Students, display an error message. When the user quits, display a report on the screen listing each group of Persons under the appropriate heading of "College Employees," "Faculty," or "Students." If the user has not entered data for one or more types of Persons during a session, display an appropriate message under the appropriate heading. Save the files as Person.java, CollegeEmployee.java, Faculty.java, Student.java, and CollegeList.java.

```
class Person{
        String firstName;
        String lastName;
        String address;
        int pincode;
        long phonenum;
        void setFirstName(String firstName){
                this.firstName= firstName;
        }
        void setLastName(String lastName){
                this.lastName=lastName;
```

```
	}
	void setAddress(String address){
		this.address = address;
	}
	void setPincode(int pincode){
		this.pincode = pincode;
	}
	void setphonenum(long phonenum){
		this.phonenum= phonenum;
	}
	void display(){
		System.out.println(this.firstName + " " + this.lastName + "'s " + "Address is " + this.address + "," + this.pincode + ".Mobile number is " + this.phonenum);
	}


}
```

```
class CollegeEmployee extends Person{
	int sn;
	double a;
	String dept;
	void setSnum(int n){
		this.sn=n;
	}
	void setAsal(double b){
		this.a=b;
	}
	void setdept(String dept){
		this.dept=dept;
	}
```

```
        void display(){
                super.display();
                System.out.println("His Security number is "+this.sn+" and Annual salary is
"+this.a+". He belongs to Department of "+this.dept+".");
        }

}
```

```
class Faculty extends CollegeEmployee
{
   boolean tenured;
   void setTenured(boolean t){
      this.tenured=t;
   }

   public void display()
   {
      super.display();
      if(tenured){
         System.out.println("Faculty member is tenured");
      }
      else{
         System.out.println("Faculty member is not tenured");
      }

   }
}
```

```
class Student extends Person
{
   String major;
```

```
    double avg;
    void setMajor(String major)
    {
        this.major = major;
    }

    void setAvg(double avg)
    {
        this.avg = avg;
    }

    public String getMajor()
    {
        return major;
    }

    public double getAvg()
    {
        return avg;
    }

    public void display()
    {
        super.display();
        System.out.println("His major is " + getMajor() + " and his average is " + getAvg());
    }
}
```

```
import java.util.Scanner;
public class CollegeList
{
    public static void main(String[] args)
```

```
  {
     CollegeEmployee[] c = new CollegeEmployee[4];
     Faculty[] f = new Faculty[3];
     Student[] s = new Student[7];
     Scanner sc = new Scanner(System.in);
     String response,fname,lname,address,dept,major;
     int pin,securitynum;
     double salary,avg;
     long phone;
     boolean tenured;
     String cont ="y";
     String QUIt = "Q";
      int i=0;
      int j=0;
      int k=0;
     System.out.println("Enter C for CollegeEmployee entry or F fot Faculty entry or S for Student
entry");
     response = sc.nextLine();
     while(response!="Q"){
        if(response.equals("C")){
        while(cont.equals("y"))
        {
           c[i]=new CollegeEmployee();
           System.out.println("Enter first name");
           fname = sc.nextLine();
           c[i].setFirstName(fname);
           System.out.println("Enter last name");
           lname = sc.nextLine();
           c[i].setLastName(lname);
           System.out.println("Enter address");
           address = sc.nextLine();
```

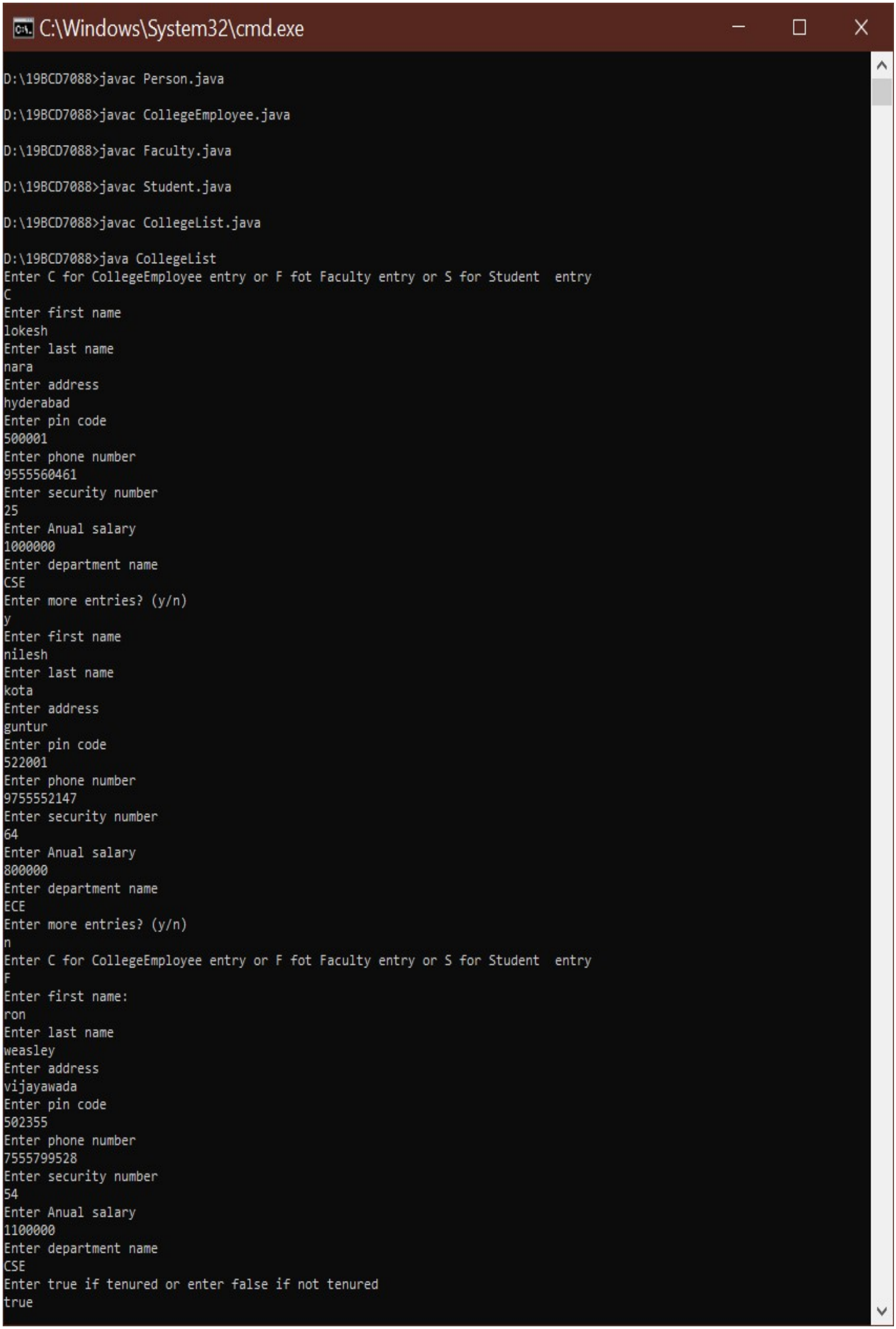
```
            c[i].setAddress(address);
            System.out.println("Enter pin code");
            pin = sc.nextInt();
            c[i].setPincode(pin);
            System.out.println("Enter phone number");
            phone = sc.nextLong();
            c[i].setphonenum(phone);
            System.out.println("Enter security number");
            securitynum = sc.nextInt();
            c[i].setSnum(securitynum);
            System.out.println("Enter Anual salary");
            salary = sc.nextDouble();
            sc.nextLine();
            c[i].setAsal(salary);
            System.out.println("Enter department name");
            dept = sc.nextLine();
            c[i].setdept(dept);
            System.out.println("Enter more entries? (y/n)");
            cont = sc.nextLine();
            if(i==3){
               System.out.println("Entered maxium number of entries");
               cont="n";
            }
            i++;
         }
        System.out.println("Enter C for CollegeEmployee entry or F fot Faculty entry or S for Student
entry");
         response = sc.nextLine();
         cont ="y";
         }
         if(response.equals("F")){
```

```
while(cont.equals("y"))
{
   f[j]=new Faculty();
   System.out.println("Enter first name: ");
   fname = sc.nextLine();
   f[j].setFirstName(fname);
   System.out.println("Enter last name");
   lname = sc.nextLine();
   f[j].setLastName(lname);
   System.out.println("Enter address");
   address = sc.nextLine();
   f[j].setAddress(address);
   System.out.println("Enter pin code");
   pin = sc.nextInt();
   f[j].setPincode(pin);
   System.out.println("Enter phone number");
   phone = sc.nextLong();
   f[j].setphonenum(phone);
   System.out.println("Enter security number");
   securitynum = sc.nextInt();
   f[j].setSnum(securitynum);
   System.out.println("Enter Anual salary");
   salary = sc.nextDouble();
   f[j].setAsal(salary);
   System.out.println("Enter department name");
   dept = sc.nextLine();
   sc.nextLine();
   f[j].setdept(dept);
   System.out.println("Enter true if tenured or enter false if not tenured");
   tenured=sc.nextBoolean();
   sc.nextLine();
```

```
            f[j].setTenured(tenured);
            System.out.println("Enter more entries ? (y/n)");
            cont = sc.nextLine();
            if(j==2){
               System.out.println("Entered maxium number of entries");
               cont="n";
            }
            j++;
         }
         System.out.println("Enter C for CollegeEmployee entry or F fot Faculty entry or S for Student entry");
         response = sc.nextLine();
         cont ="y";
         }
         if(response.equals("S")){
         while(cont.equals("y"))
         {
            s[k]=new Student();
            System.out.println("Enter first name");
            fname = sc.nextLine();
            s[k].setFirstName(fname);
            System.out.println("Enter last name");
            lname = sc.nextLine();
            s[k].setLastName(lname);
            System.out.println("Enter address");
            address = sc.nextLine();
            s[k].setAddress(address);
            System.out.println("Enter pin code");
            pin = sc.nextInt();
            s[k].setPincode(pin);
            System.out.println("Enter phone number");
```

```
        phone = sc.nextLong();
        sc.nextLine();
        s[k].setphonenum(phone);
        System.out.println("Enter major");
        major =sc.nextLine();
        s[k].setMajor(major);
        System.out.println("Enter average");
        avg=sc.nextDouble();
        sc.nextLine();
        s[k].setAvg(avg);
        System.out.println("Enter more entries? (y/n)");
        cont = sc.nextLine();
        if(k==6){
            System.out.println("Entered maxium number of entries");
            cont="n";
        }
        k++;
    }
    System.out.println("quiting");
    response="Q";



    }
}
if(response.equals("Q")){

    for (int p =0;p<i;p++){
        c[p].display();
    }
    for(int q=0;q<j;q++){
        f[q].display();
```

```
            }
            for(int r=0;r<k;r++){
                s[r].display();
            }
        }
        int h = 4-i;
        if(h!=0){
        System.out.println(h +" CollegeEmployee data not Entered");
    }
        h=3-j;
        if(h!=0){
        System.out.println(h+ " Faculty data not Entered");
    }
        h=7-k;
        if(h!=0){
        System.out.println(h + " Student data not Entered");
    }

    }
}
```


C:\Windows\System32\cmd.exe
D:\19BCD7088>javac Person.java
D:\19BCD7088>javac CollegeEmployee.java
D:\19BCD7088>javac Faculty.java
D:\19BCD7088>javac Student.java
D:\19BCD7088>javac CollegeList.java
D:\19BCD7088>java CollegeList
Enter C for CollegeEmployee entry or F fot Faculty entry or S for Student entry
C
Enter first name
lokesh
Enter last name
nara
Enter address
hyderabad
Enter pin code
500001
Enter phone number
9555560461
Enter security number
25
Enter Anual salary
1000000
Enter department name
CSE
Enter more entries? (y/n)
y
Enter first name
nilesh
Enter last name
kota
Enter address
guntur
Enter pin code
522001
Enter phone number
9755552147
Enter security number
64
Enter Anual salary
800000
Enter department name
ECE
Enter more entries? (y/n)
n
Enter C for CollegeEmployee entry or F fot Faculty entry or S for Student entry
F
Enter first name:
ron
Enter last name
weasley
Enter address
vijayawada
Enter pin code
502355
Enter phone number
7555799528
Enter security number
54
Enter Anual salary
1100000
Enter department name
CSE
Enter true if tenured or enter false if not tenured
true

```
true
Enter more entries ? (y/n)
n
Enter C for CollegeEmployee entry or F fot Faculty entry or S for Student  entry
S
Enter first name
bhuvanesh
Enter last name
valiveti
Enter address
agiripalli
Enter pin code
521211
Enter phone number
9014914993
Enter major
CSE
Enter average
8.89
Enter more entries? (y/n)
n
quiting
lokesh nara's Address is hyderabad,500001.Mobile number is 9555560461
His Security number is 25 and Annual salary is 1000000.0. He belongs to Department of CSE.
nilesh kota's Address is guntur,522001.Mobile number is 9755552147
His Security number is 64 and Annual salary is 800000.0. He belongs to Department of ECE.
ron weasley's Address is vijayawada,502355.Mobile number is 7555799528
His Security number is 54 and Annual salary is 1100000.0. He belongs to Department of .
Faculty member is tenured
bhuvanesh valiveti's Address is agiripalli,521211.Mobile number is 9014914993
His major is CSE and his average is 8.89
2 CollegeEmployee data not Entered
2 Faculty data not Entered
6 Student data not Entered

D:\19BCD7088>
```

Lab4

L1 slot

V.Manikanta Bhuvanesh

19BCD7088

1.a.

Create an abstract class named Book. Include a String field for the book'stitle and a double field for the book's price. Within the class, include aconstructor that requires the book title, and add two get methods—one that returns the title and one that returns the price. Include an abstract method named setPrice(). Create two child classes of Book: Fiction and NonFiction. Each must include a setPrice() method that sets the price for all Fiction Books to $24.99 and for all NonFiction Books to $37.99. Write a constructor for each subclass, and include a call to setPrice() within each. Write an application demonstrating that you can create both a Fiction and a NonFiction Book, and display their fields. Save the files as Book.java, Fiction.java, NonFiction.java, and UseBook.java.

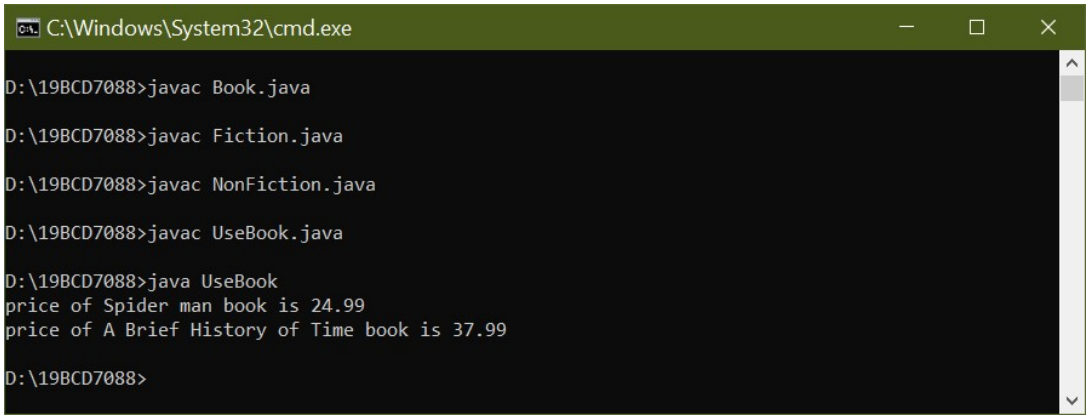
```
abstract class Book {
   String t;
   double p;

   Book(String title ){
      t=title;
   }

   String gettitle(){
      return t;
   }

   double getPrice(){
      return p;
   }

   abstract void setPrice();
}
```

```
class Fiction extends Book{

   Fiction(String title) {
      super(title);
      setPrice();
```

```
    }

    void setPrice(){
        super.p=24.99;
    }
}
```

```
class NonFiction extends Book{

    NonFiction(String title) {
        super(title);
        setPrice();
    }

    void setPrice(){
        super.p=37.99;
    }
}
```

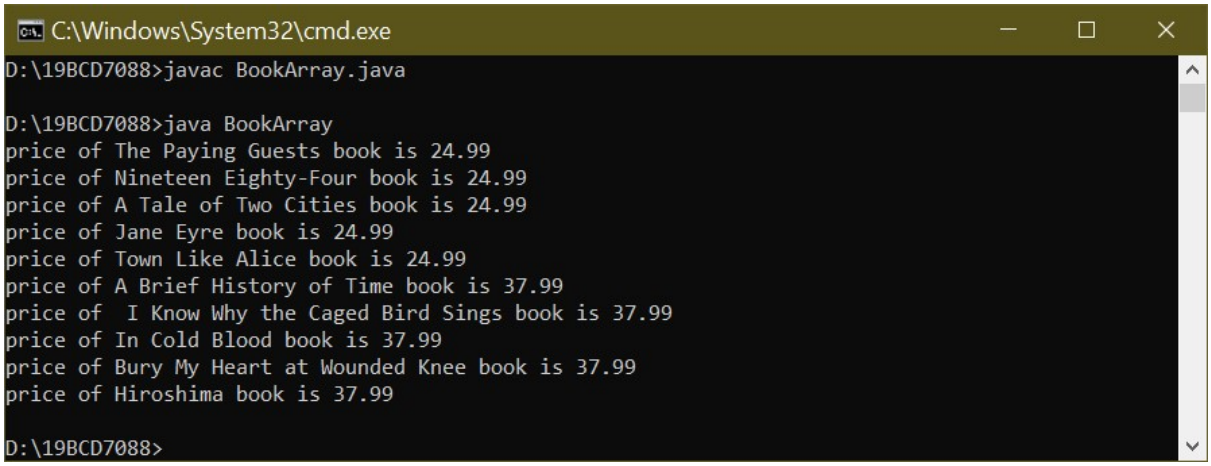
```
public class UseBook {
    public static void main(String[] args){
        Book b1,b2;
        b1=new Fiction("Spider man");
        System.out.println("price of " + b1.gettitle() + " book is " + b1.getPrice());
        b2=new NonFiction("A Brief History of Time");
        System.out.println("price of " + b2.gettitle() + " book is " + b2.getPrice());
    }
}
```



1.b.

Write an application named BookArray in which you create an array that holds 10 Books, some Fiction and some NonFiction. Using a for loop, display details about all 10 books. Save the file as BookArray.java.

```
public class BookArray{
	public static void main(String[] args) {
		Book b[] = new Book[10];
		b[0] = new Fiction("The Paying Guests");
		b[1] = new Fiction("Nineteen Eighty-Four");
		b[2] = new Fiction("A Tale of Two Cities");
		b[3] = new Fiction("Jane Eyre");
		b[4] = new Fiction("Town Like Alice");
		b[5] = new NonFiction("A Brief History of Time");
		b[6] = new NonFiction(" I Know Why the Caged Bird Sings");
		b[7] = new NonFiction("In Cold Blood");
		b[8] = new NonFiction("Bury My Heart at Wounded Knee");
		b[9] = new NonFiction("Hiroshima");
		for(int i=0;i<10;i++){
			System.out.println("price of " + b[i].gettitle() + " book is " + b[i].getPrice());
		}
	}
}
```



2.a.

The Talk-A-Lot Cell Phone Company provides phone services for its customers. Create an abstract class named PhoneCall that includes a String field for a phone number and a double field for the price of the call. Also include a constructor that requires a phone number parameter and that sets the price to 0.0. Include a set method for the price. Also include three abstract get methods—one that returns the phone number, another that returns the price of the call, and a third that displays information about the call. Create two child classes of PhoneCall: IncomingPhoneCall and OutgoingPhoneCall. The IncomingPhoneCall constructor passes its phone number parameter to its parent's constructor and sets the price of the call to 0.02. The method that displays the phone call information displays the phone number, the rate, and the price of the call (which is the same as the rate). The OutgoingPhoneCall class includes an additional field that holds the time of the call in minutes. The constructor requires both a phone number and the time. The price is 0.04 per minute, and the display method shows the details of the call, including the phone number, the rate per minute, the number of minutes, and the total price. Write an application that demonstrates you can instantiate and display both IncomingPhoneCall and OutgoingPhoneCall objects. Save the files as PhoneCall.java, IncomingPhoneCall.java, OutgoingPhoneCall.java, and DemoPhoneCalls.java.

```
abstract class PhoneCall
{
   String phn;
   double price;
    PhoneCall(String phn)
   {
      this.phn =  phn;
      this.price = 0.0;
   }
   abstract String getPhoneNumber();
   abstract double getPrice();
```

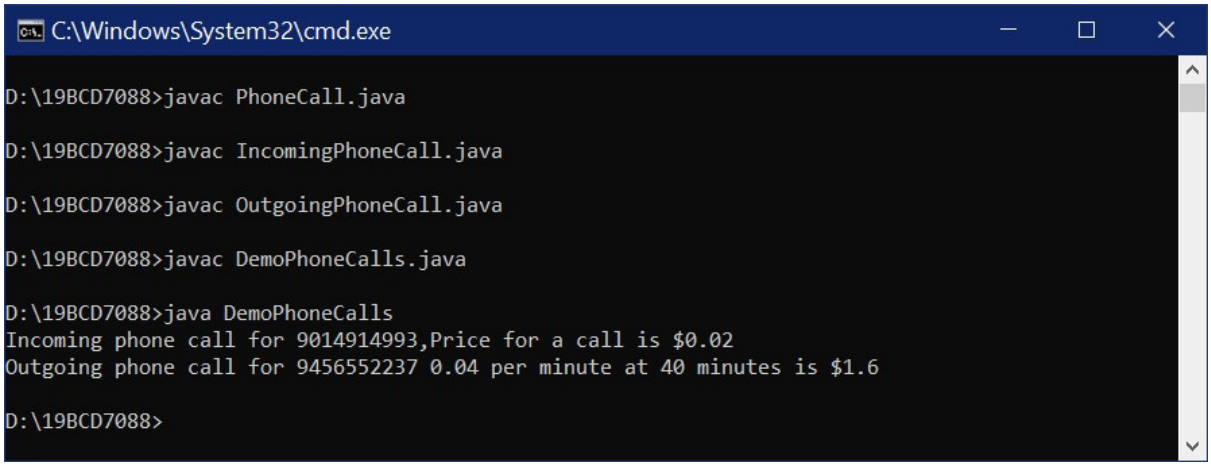
```
    abstract void getInf();
    abstract void setPrice();

}
```

```
class IncomingPhoneCall extends PhoneCall {

    double r=0.02;
     IncomingPhoneCall(String phoneNumber){
        super(phoneNumber);
        setPrice();
    }
    void setPrice() {
        super.price = 0.02;
    }
     void getInf(){
        System.out.println("Incoming phone call for "+getPhoneNumber()+",Price for a call is
$"+getPrice());
    }
      String getPhoneNumber()
      {
         return super.phn;
      }
      double getPrice()
      {
         return super.price;
      }
}
```

```
class OutgoingPhoneCall extends PhoneCall {

    double r = 0.04;
    int minutes;
```

```
    OutgoingPhoneCall(String phoneNumber, int minutes){
      super(phoneNumber);
      this.minutes = minutes;
      setPrice();
   }
   void setPrice() {
      super.price = 0.04;
   }
   void getInf() {
      System.out.println("Outgoing phone call for " + getPhoneNumber() + " "+ r + " per minute at " +
minutes + " minutes is $" + price*minutes);
   }
    public String getPhoneNumber()
    {
       return super.phn;
    }
    public double getPrice()
    {
       return super.price;
    }

}
```

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

   IncomingPhoneCall in=new IncomingPhoneCall("9014914993");
   OutgoingPhoneCall out=new OutgoingPhoneCall("9456552237",40);
   in.getInf();
   out.getInf();
 }
}
```

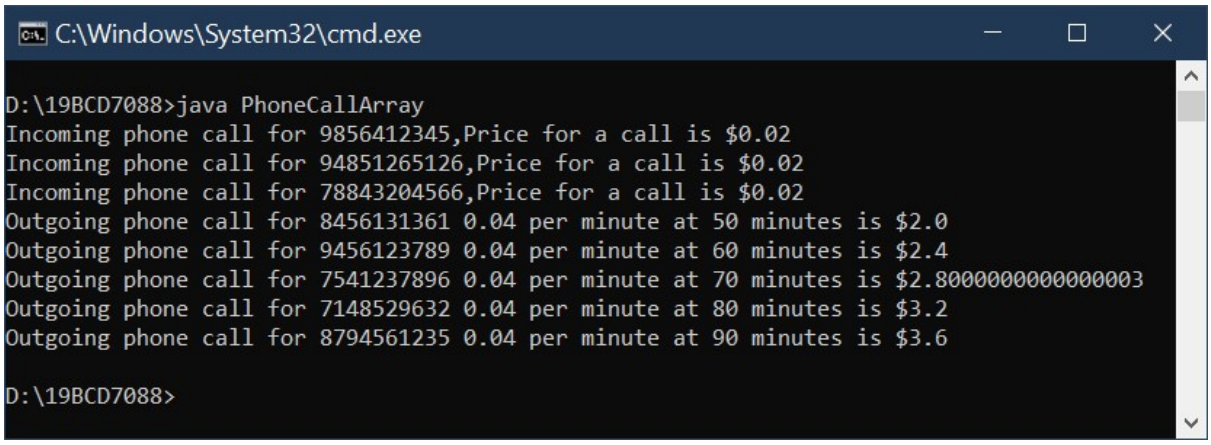


2.b.

Write an application in which you assign data to a mix of eight IncomingPhoneCall and OutgoingPhoneCall objects into an array. Use a for loop to display the data. Save the file as PhoneCallArray.java.

```
public class PhoneCallArray{
        public static void main(String[] args) {
                IncomingPhoneCall in[]=new IncomingPhoneCall[3];
                OutgoingPhoneCall out[]=new OutgoingPhoneCall[5];
                in[0]=new IncomingPhoneCall("9856412345");
                in[1]=new IncomingPhoneCall("94851265126");
                in[2]=new IncomingPhoneCall("78843204566");
                out[0]= new OutgoingPhoneCall("8456131361",50);
                out[1]= new OutgoingPhoneCall("9456123789",60);
                out[2]= new OutgoingPhoneCall("7541237896",70);
                out[3]= new OutgoingPhoneCall("7148529632",80);
                out[4]= new OutgoingPhoneCall("8794561235",90);
                for(int i=0;i<3;i++){
                        in[i].getInf();
                }
                for(int j=0;j<5;j++){
                        out[j].getInf();
                }
        }
```

}



3.a.

Create an interface named Turner, with a single method named turn(). Create a class named Leaf that implements turn() to display Changing colors. Create a class named Page that implements turn() to display Going to the next page. Create a class named Pancake that implements turn() to display Flipping. Write an application named DemoTurners that creates one object of each of these class types and demonstrates the turn() method for each class. Save the files as Turner.java, Leaf.java, Page.java, Pancake.java, and DemoTurners.java.

```
interface Turner {
   public void turn();
}
```

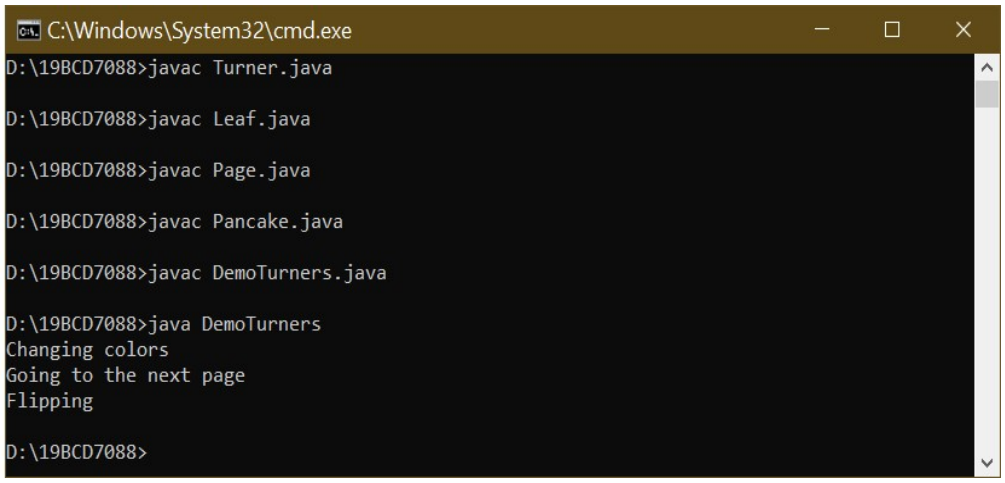
```
class Leaf implements Turner{
   public void turn() {
      System.out.println("Changing colors");
   }
}
```

```
class Page implements Turner{
   public void turn() {
      System.out.println("Going to the next page");
   }
}
```

```
class Pancake implements Turner{
   public void turn() {
```

```
        System.out.println("Flipping");
    }
}
```

```
public class DemoTurners {
    public static void main(String[] args){
        Leaf l = new Leaf();
        Page p=new Page();
        Pancake c=new Pancake();
        l.turn();
        p.turn();
        c.turn();
    }
}
```



3.b.

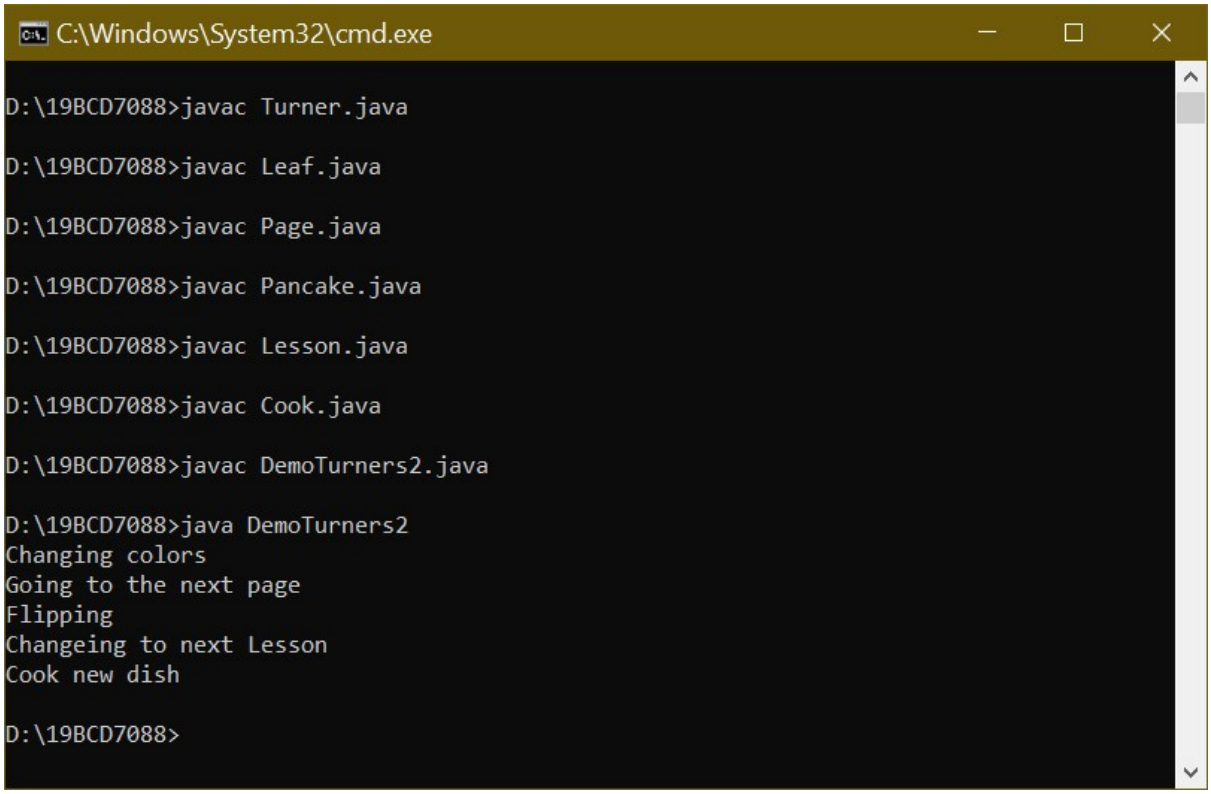
Think of two more objects that use turn(), create classes for them, and then add objects to the DemoTurners application, renaming it DemoTurners2. java. Save the files, using the names of new objects that use turn().

```
interface Turner {
    public void turn();
}
```

```
class Leaf implements Turner{
    public void turn() {
```

```
        System.out.println("Changing colors");
    }
}
class Page implements Turner{
    public void turn() {
        System.out.println("Going to the next page");
    }
}
class Pancake implements Turner{
    public void turn() {
        System.out.println("Flipping");
    }
}
class Lesson implements Turner{
    public void turn() {
        System.out.println("Changeing to next Lesson");
    }
}
class Cook implements Turner{
    public void turn() {
        System.out.println("Cook new dish");
    }
}
public class DemoTurners2 {
    public static void main(String[] args){
        Leaf l = new Leaf();
        Page p=new Page();
        Pancake c=new Pancake();
        Lesson le = new Lesson();
        Cook co = new Cook();
```

```
        l.turn();
        p.turn();
        c.turn();
        le.turn();
        co.turn();
    }
}
```



3.c.

Apply Dynamic method dispatch to show the power of it and name the class as DemoTurners3.

```
interface Turner {
    public void turn();
}
```

```
class Leaf implements Turner{
    public void turn() {
        System.out.println("Changing colors");
    }
```

```
}
```

```
class Page implements Turner{
   public void turn() {
      System.out.println("Going to the next page");
   }
}
```

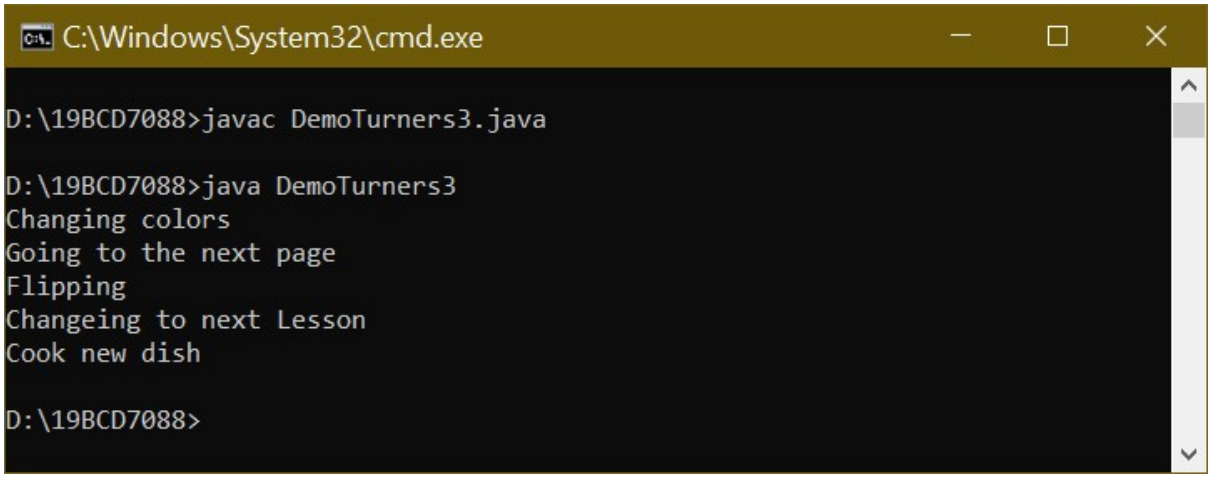
```
class Pancake implements Turner{
   public void turn() {
      System.out.println("Flipping");
   }
}
```

```
class Lesson implements Turner{
   public void turn() {
      System.out.println("Changeing to next Lesson");
   }
}
```

```
class Cook implements Turner{
   public void turn() {
      System.out.println("Cook new dish");
   }
}
```

```
public class DemoTurners3 {
   public static void main(String[] args){
      Turner t;
      t = new Leaf();
      t.turn();
      t=new Page();
      t.turn();
      t=new Pancake();
      t.turn();
```

```
        t= new Lesson();
        t.turn();
        t = new Cook();
        t.turn();
    }
}
```



4.a.

Create an abstract class called GeometricFigure. Each figure includes a height, a width, a figure type, and an area. Include an abstract method to determine the area of the figure. Create two subclasses called Square and Triangle. Create an application that demonstrates creating objects of both subclasses, and store them in an array. Save the files as GeometricFigure.java, Square.java, Triangle.java, and UseGeometric.java.

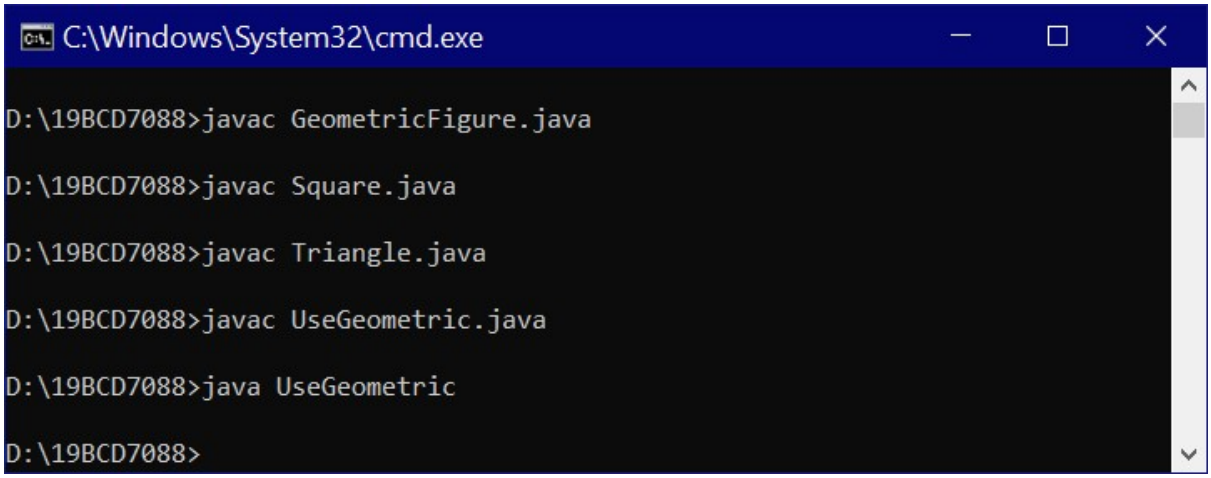
```
abstract class GeometricFigure {
    int height, width;
    String figureType;
    int area;
    abstract void Area(int h, int w);
}
```

```
class Square extends GeometricFigure{
    Square(int a, int b){
        super.height=a;
        super.width=b;
```

```
        Area(a,b);
    }
    void Area(int h, int w) {
      super.area=(h*w);
    }
}
```

```
class Triangle extends GeometricFigure{

    Triangle(int a, int b){
        super.height=a;
        super.width=b;
        Area(a,b);
    }
    void Area(int h, int w) {
        area=(h*w)/2;
    }

}
```

```
public class UseGeometric {
    public static void main(String[] args){
        GeometricFigure f[]=new GeometricFigure[2];
        Square s=new Square(20,20);
        Triangle t=new Triangle(10,20);
        f[0]=s;
        f[1]=t;
    }

}
```



4.b.

Modify 4.a., adding an interface called SidedObject that contains a method called displaySides(); this method displays the number of sides the object possesses. Modify the GeometricFigure subclasses to include the use of the interface to display the number of sides of the figure. Create an application that demonstrates the use of both subclasses. Save the files as GeometricFigure2.java, Square2.java, Triangle2.java, SidedObject.java, and UseGeometric2.java.

```
abstract class GeometricFigure {
   int height, width;
   String figureType;
   int area,sides;
   abstract void Area(int h, int w);
}
```

```
interface SidedObject{
        public void display();
}
```

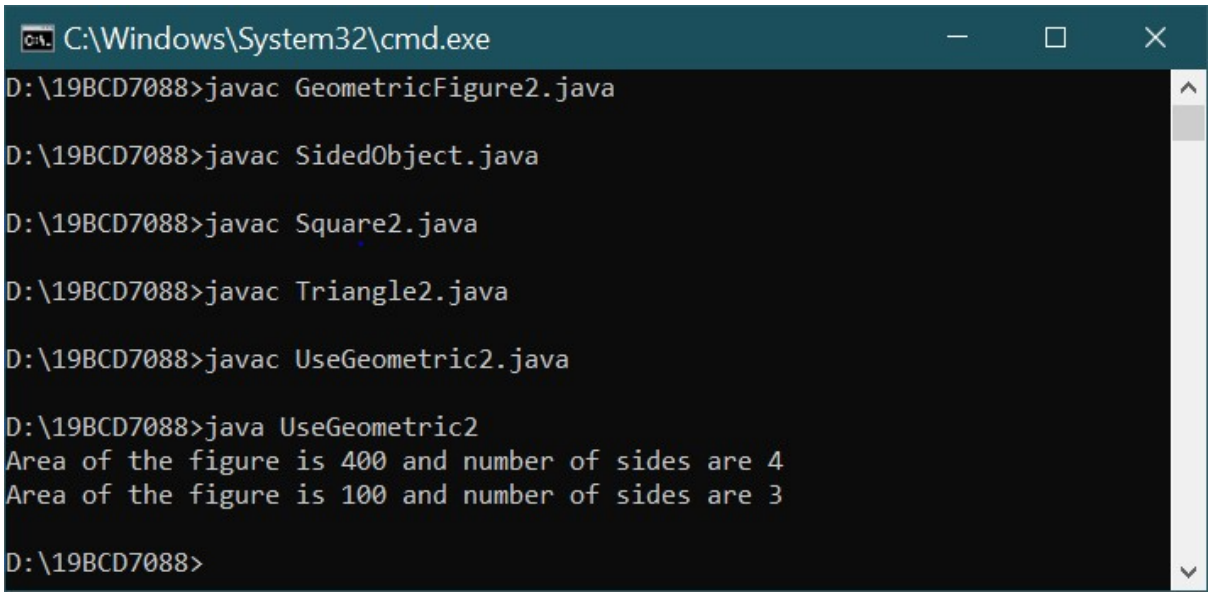
```
class Square2 extends GeometricFigure implements SidedObject{
   Square2(int a, int b){
      super.height=a;
      super.width=b;
      super.sides=4;
      Area(a,b);
   }
   void Area(int h, int w) {
```

```
     super.area=(h*w);
   }
   public void display(){
         System.out.println("Area of the figure is " + super.area + " and number of sides are " +
super.sides);
   }
}
```

```
class Triangle2 extends GeometricFigure implements SidedObject{

   Triangle2(int a, int b){
      super.height=a;
      super.width=b;
      super.sides=3;
      Area(a,b);
   }
   void Area(int h, int w) {
      super.area=(h*w)/2;
   }
   public void display(){
         System.out.println("Area of the figure is " + super.area + " and number of sides are " +
super.sides);
   }
}
```

```
public class UseGeometric2 {
   public static void main(String[] args){
      Square2 s=new Square2(20,20);
      Triangle2 t=new Triangle2(10,20);
      s.display();
      t.display();
   }
```

}



5.

Sanchez Construction Loan Co. makes loans of up to $100,000 for construction projects. There are two categories of Loans—those to businesses and those to individual applicants.

Write an application that tracks all new construction loans. The application also must calculate the total amount owed at the due date (original loan amount + loan fee). The application should include the following classes:

•• Loan—A public abstract class that implements the LoanConstants interface. A Loan includes a loan number, customer last name, amount of loan, interest rate, and term. The constructor requires data for each of the fields except interest rate. Do not allow loan amounts greater than $100,000. Force any loan term that is not one of the three defined in the LoanConstants class to a short-term, 1-year loan. Create a toString() method that displays all the loan data.

•• LoanConstants—A public interface class. LoanConstants includes constant values for short-term (1 year), medium-term (3 years), and long-term (5 years) loans. It also contains constants for the company name and the maximum loan amount.

•• BusinessLoan—A public class that extends Loan. The BusinessLoan constructor sets the interest rate to 1% more than the current prime interest rate.

•• PersonalLoan—A public class that extends Loan. The PersonalLoan constructor sets the interest rate to 2% more than the current prime interest rate.

•• CreateLoans—An application that creates an array of five Loans. Prompt the user for the current prime interest rate. Then, in a loop, prompt the user for a loan type and all relevant information for that loan. Store the created Loan objects in the array. When data entry is complete, display all the loans.

Save the files as Loan.java, LoanConstants.java, BusinessLoan.java, PersonalLoan.java, and CreateLoans.java.

```
interface LoanConstants {
   public int st = 1;
   public int mt = 3;
   public int lt = 5;
   public String cn = "Sanchez Construction Loan Co.";
   public double max = 100000;
}
```

```
abstract class Loan implements LoanConstants {
   String loanNum;
   String lastName;
   double loanAmt;
   double interestRate;
   int term;
   Loan(String loanNum, String lastName, double loanAmt, int term) {
      this.loanNum = loanNum;
      this.lastName = lastName;
      if (loanAmt > max) {
            System.out.println("Loan amount value is more than $100,000");
         }
      else {
            this.loanAmt = loanAmt;
         }
      if(term==st||term==mt||term==lt){
         this.term=term;
      }
```

```
        else{
            this.term=1;
        }

    }
    public String toString() {
        double n =this.loanAmt+(this.loanAmt * (this.interestRate/100));
        return this.lastName + "'s loan number is " + this.loanNum + " his loan amount is " +
this.loanAmt + " with intrest rate of " + this.interestRate +" and total due is " + n + " in term " +
this.term;
    }
}
```

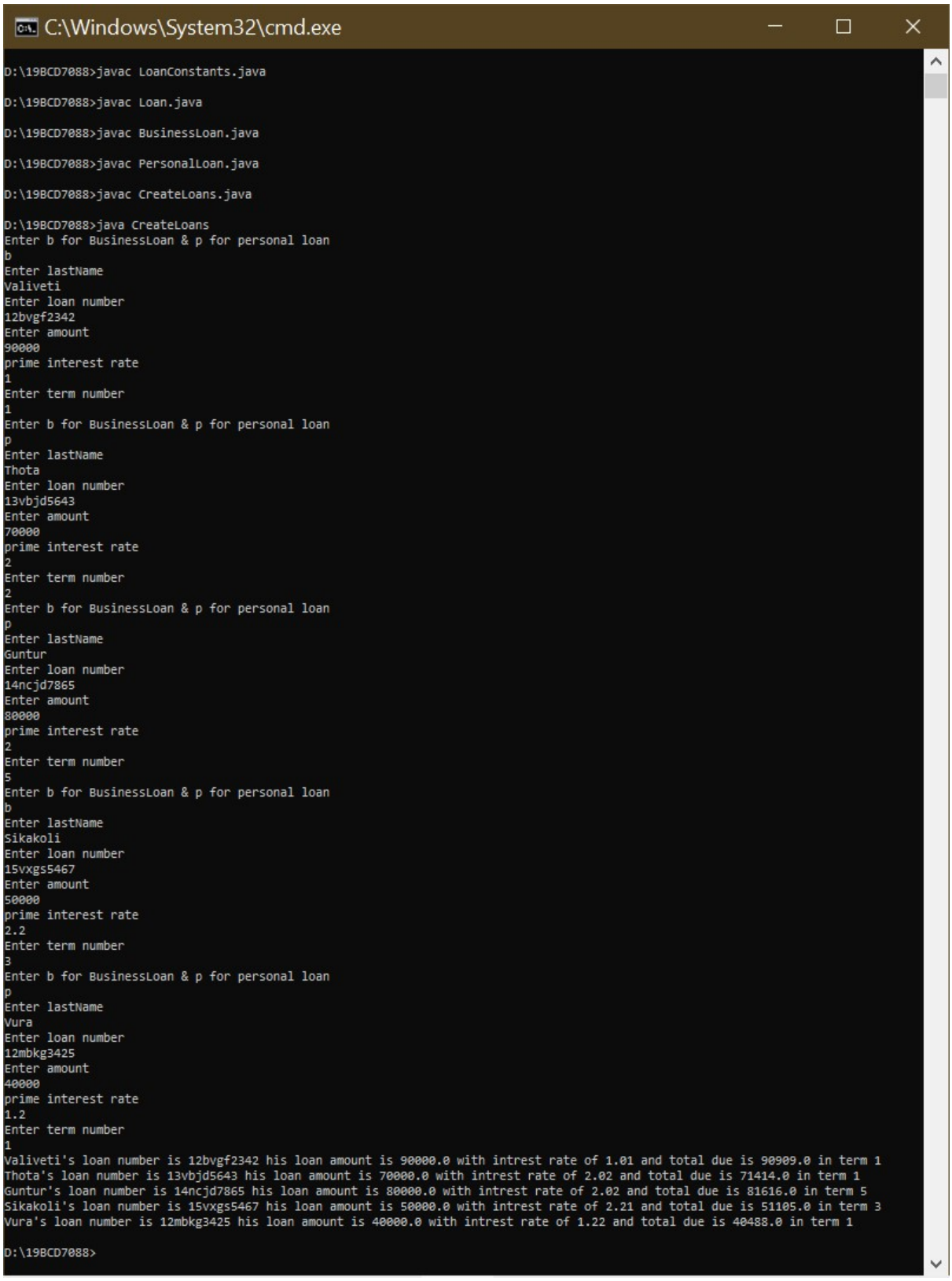
```
class BusinessLoan extends Loan {
    BusinessLoan(String loanNum, String lastName, double loanAmt, int term, double primeIntRate) {
        super(loanNum, lastName, loanAmt, term);
        super.interestRate = 0.01 +primeIntRate;
    }
}
```

```
class PersonalLoan extends Loan {
    PersonalLoan(String loanNum, String lastName, double loanAmt, int term, double primeIntRate) {
        super(loanNum, lastName, loanAmt, term);
        super.interestRate = 0.02 + primeIntRate ;
    }
}
```

```
import java.util.Scanner;
public class CreateLoans{
            public static void main(String[] args) {
                        Scanner sc = new Scanner(System.in);
                        Loan[] l = new Loan[5];
                        String ch,ln,lon,p;
                        double amt,ir;
```

```
int term;
for(int i=0;i<5;i++){
   System.out.println("Enter b for BusinessLoan & p for personal loan");
   if(i!=0){
      sc.nextLine();
   }
          ch=sc.nextLine();
          System.out.println("Enter lastName");
          ln=sc.nextLine();
          System.out.println("Enter loan number");
          lon=sc.nextLine();
          System.out.println("Enter amount");
          amt=sc.nextDouble();
          System.out.println("prime interest rate");
          ir=sc.nextDouble();
          System.out.println("Enter term number");
          term=sc.nextInt();
          if(ch.equals("b")){
                    l[i]=new BusinessLoan(lon,ln,amt,term,ir);
          }
          else if(ch.equals("p")){
                    l[i]=new PersonalLoan(lon,ln,amt,term,ir);
          }
          else{
                    System.out.println("Invalid loan type");
          }
}
for(int j = 0;j<5;j++){
          p=l[j].toString();
          System.out.println(p);
}
```

```
        }
}
```


C:\Windows\System32\cmd.exe
D:\19BCD7088>javac LoanConstants.java
D:\19BCD7088>javac Loan.java
D:\19BCD7088>javac BusinessLoan.java
D:\19BCD7088>javac PersonalLoan.java
D:\19BCD7088>javac CreateLoans.java
D:\19BCD7088>java CreateLoans
Enter b for BusinessLoan & p for personal loan
b
Enter lastName
Valiveti
Enter loan number
12bvgf2342
Enter amount
90000
prime interest rate
1
Enter term number
1
Enter b for BusinessLoan & p for personal loan
p
Enter lastName
Thota
Enter loan number
13vbjd5643
Enter amount
70000
prime interest rate
2
Enter term number
2
Enter b for BusinessLoan & p for personal loan
p
Enter lastName
Guntur
Enter loan number
14ncjd7865
Enter amount
80000
prime interest rate
2
Enter term number
5
Enter b for BusinessLoan & p for personal loan
b
Enter lastName
Sikakoli
Enter loan number
15vxgs5467
Enter amount
50000
prime interest rate
2.2
Enter term number
3
Enter b for BusinessLoan & p for personal loan
p
Enter lastName
Vura
Enter loan number
12mbkg3425
Enter amount
40000
prime interest rate
1.2
Enter term number
1
Valiveti's loan number is 12bvgf2342 his loan amount is 90000.0 with intrest rate of 1.01 and total due is 90909.0 in term 1
Thota's loan number is 13vbjd5643 his loan amount is 70000.0 with intrest rate of 2.02 and total due is 71414.0 in term 1
Guntur's loan number is 14ncjd7865 his loan amount is 80000.0 with intrest rate of 2.02 and total due is 81616.0 in term 5
Sikakoli's loan number is 15vxgs5467 his loan amount is 50000.0 with intrest rate of 2.21 and total due is 51105.0 in term 3
Vura's loan number is 12mbkg3425 his loan amount is 40000.0 with intrest rate of 1.22 and total due is 40488.0 in term 1
D:\19BCD7088>

Lab 5

L1 slot

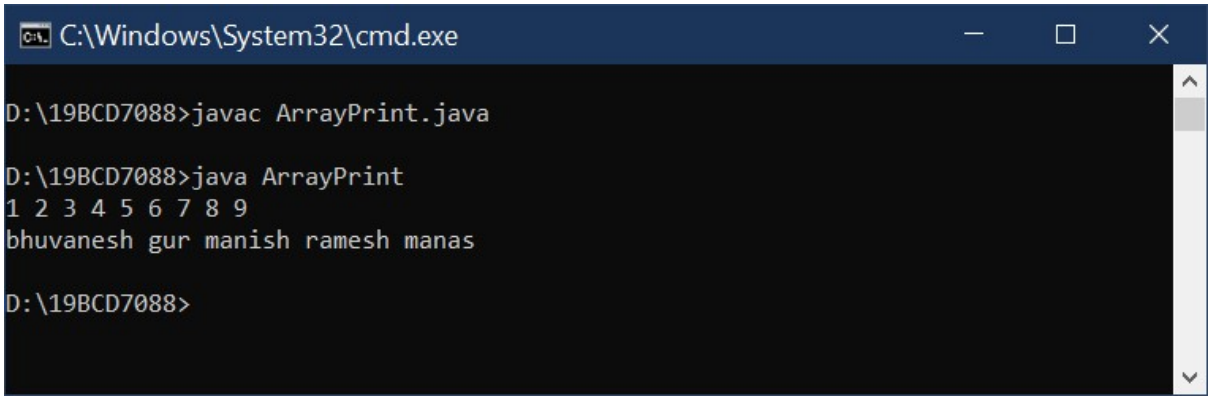
Valiveti manikanta bhuvanesh

19BCD7088

1.

Let's say you have an integer array and a string array. You have to write a single method printArray that can print all the elements of both arrays. The method should be able to accept both integer arrays or string arrays.(Do not use overloading, use generics).Name the file ArrayPrint.java

```
import java.util.*;

public class ArrayPrint{
	static <T> void printArray(T[] a){
		for(int i=0;i<a.length;i++){
			System.out.print(a[i]+" ");
		}
		System.out.println();
	}
	public static void main(String[] args) {
		Integer [] a={1,2,3,4,5,6,7,8,9};
		String [] str = {"bhuvanesh","gur","manish","ramesh","manas"};
		printArray(a);
		printArray(str);
	}
}
```

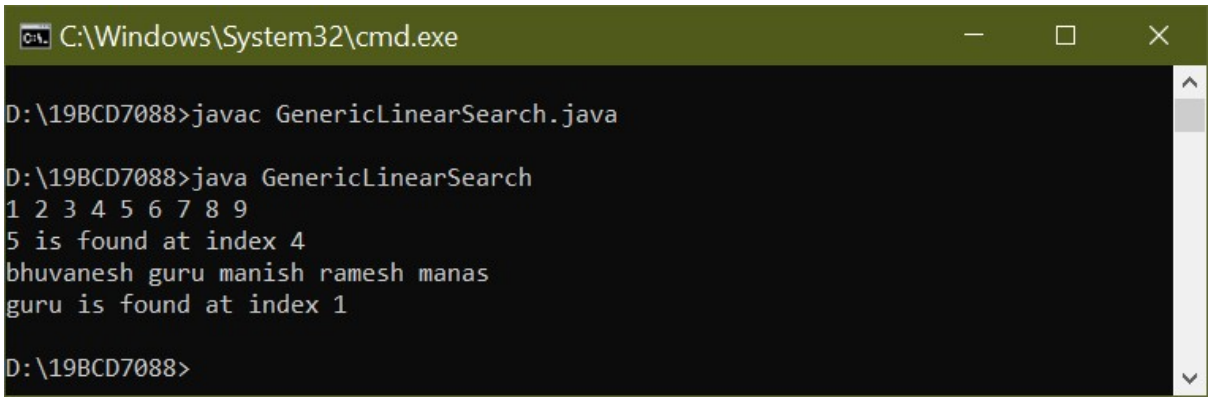


2.

Write a generic method to perform linearSearch on array of objects. Name the file GenericLinearSearch.java

```
import java.util.*;
public class GenericLinearSearch{
	static <T> int linearSearch(T[]a,T n){
		for(int i=0;i<a.length;i++){
			if(a[i].equals(n)){
				return i;
			}
		}
		return 0;
	}
	static <T> void printArray(T[] a){
		for(int i=0;i<a.length;i++){
			System.out.print(a[i]+" ");
		}
		System.out.println();
	}
	public static void main(String[] args) {
		Integer [] a={1,2,3,4,5,6,7,8,9};
		Integer k1=5;
		printArray(a);
		int k = linearSearch(a,k1);
		System.out.println(k1 + " is found at index " + k);
		String [] str = {"bhuvanesh","guru","manish","ramesh","manas"};
		String k2="guru";
		printArray(str);
		int p = linearSearch(str,k2);
		System.out.println(k2 + " is found at index " + p);
```

```
        }
}
```



3.

Write a generic class called GenericStack<T> that represents a stack structure. A stack structure follow the strategy last-in-first-out, which means that the last element added to the stack, is the first to be taken out.

The GenericStack class has the following attributes and methods:

--An attribute ArrayList<T> elements which represents the elements of the stack.(All of you refer collection framework for ArrayList. or you can use an array to hold the elements of Stack.)[Refer the following links to have intro on ArrayList:
https://www.w3schools.com/java/java_arraylist.asp,
https://www.geeksforgeeks.org/arraylist-in-java/]

--A constructor that creates the ArrayList or an Array

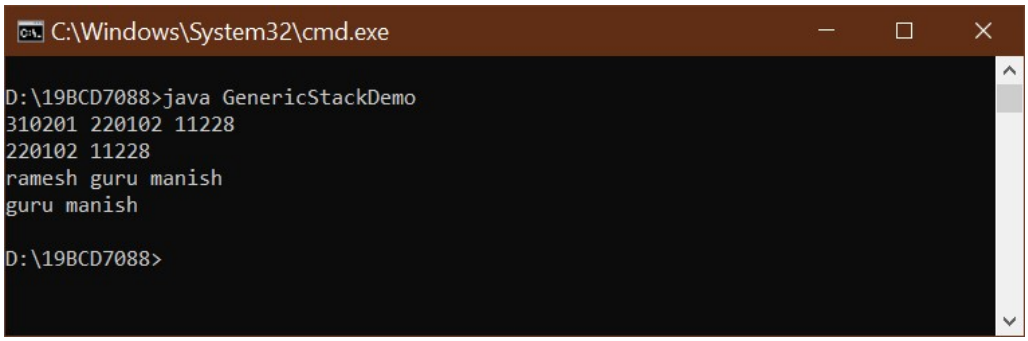
--A method push(T e) which adds the element to the ArrayList<T> or array.

--A method pop() which removes the last element of the ArrayList<T> (last element added), if the list is not already empty and returns it.

--A method print() which prints the elements of the stack starting from the last element to the first element.

```
import java.util.*;
class GenericStack<T>{
	int top;
	int max;
	T [] a;
	GenericStack(int size){
		a=(T[]) new Object[size];
		max=size;
		top=-1;
	}
	void push(T x){
		if(top==(max-1)){
			System.out.println("Stack is Filled");
		}
		else{
			top++;
			a[top]=x;
		}
	}
	T pop(){
		if(top==-1){
			System.out.println("Stack is empty");
			return null;
		}
		else{
			return a[top--];
		}
	}
	void print(){
```

```
                for (int i=top;i>=0;i--){
                        System.out.print(a[i] + " ");
                }
                System.out.println();
        }
}
public class GenericStackDemo{
        public static void main(String[] args) {
                GenericStack<Integer> s =new GenericStack<Integer>(3);
                GenericStack<String> str =new GenericStack<String>(3);
                s.push(11228);
                s.push(220102);
                s.push(310201);
                s.print();
                s.pop();
                s.print();
                str.push("manish");
                str.push("guru");
                str.push("ramesh");
                str.print();
                str.pop();
                str.print();
        }
}
```



3. b.

Create the generic interface GenericStackable<T> that contains the abstract methods:

-an abstract method push(T e) which adds the element to the ArrayList<T>

-an abstract method pop() which remove the last element of the ArrayList<T> (last element added), if the list is not already empty.

-a abstract method print() which prints the elements of the stack starting from the last element to the first element.

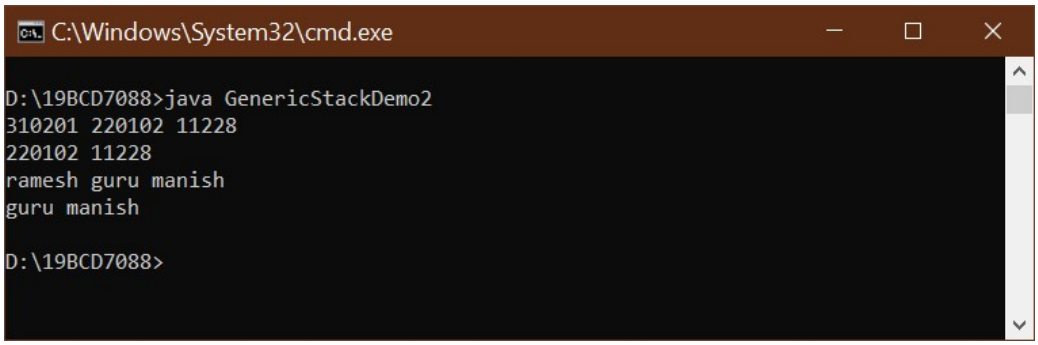
-a abstract method isEmpty() that would return true if the stack is empty, and false otherwise.

Modify the class GenericStack<T> such that it implements the generic interface GenericStackable<T> . Create a class GenericStackDemo2 and work with two different stacks.

```
import java.util.*;
interface GenericStackable<T>{
	public void push(T x);
	public T pop();
	public void print();
	public boolean isEmpty();
}
class GenericStack<T> implements GenericStackable<T> {
	int top;
	int max;
	T [] a;
	GenericStack(int size){
		a=(T[]) new Object[size];
		max=size;
```

```
        top=-1;
}
public void push(T x){
        if(top==(max-1)){
                System.out.println("Stack is Filled");
        }
        else{
                top++;
                a[top]=x;
        }
}
public T pop(){
        if(top==-1){
                System.out.println("Stack is empty");
                return null;
        }
        else{
                return a[top--];
        }
}
public void print(){
        for (int i=top;i>=0;i--){
                System.out.print(a[i] + " ");
        }
        System.out.println();
}
public boolean isEmpty(){
        if(top==-1){
                return true;
```

```
                }
                else{
                        return false;
                }
        }
}
public class GenericStackDemo2{
        public static void main(String[] args) {
                GenericStack<Integer> s =new GenericStack<Integer>(3);
                GenericStack<String> str =new GenericStack<String>(3);
                s.push(11228);
                s.push(220102);
                s.push(310201);
                s.print();
                s.pop();
                s.print();
                str.push("manish");
                str.push("guru");
                str.push("ramesh");
                str.print();
                str.pop();
                str.print();
        }
}
```



Lab 6

L1 slot

Valiveti manikanta bhuvanesh

19BCD7088

1.

Create the generic interface GenericQueuable<T> that contains the following abstract methods:

- an abstract method insertEnd(T e) which adds the element to the Queue at end.

- an abstract method removeBegin() which removes a element from the begining of the queue.

- an abstract method printQueue() which prints the queue elements from the front of the queue to end.

- an abstract method isQueueEmpty() which returns true if the queue is empty otherwise return false.

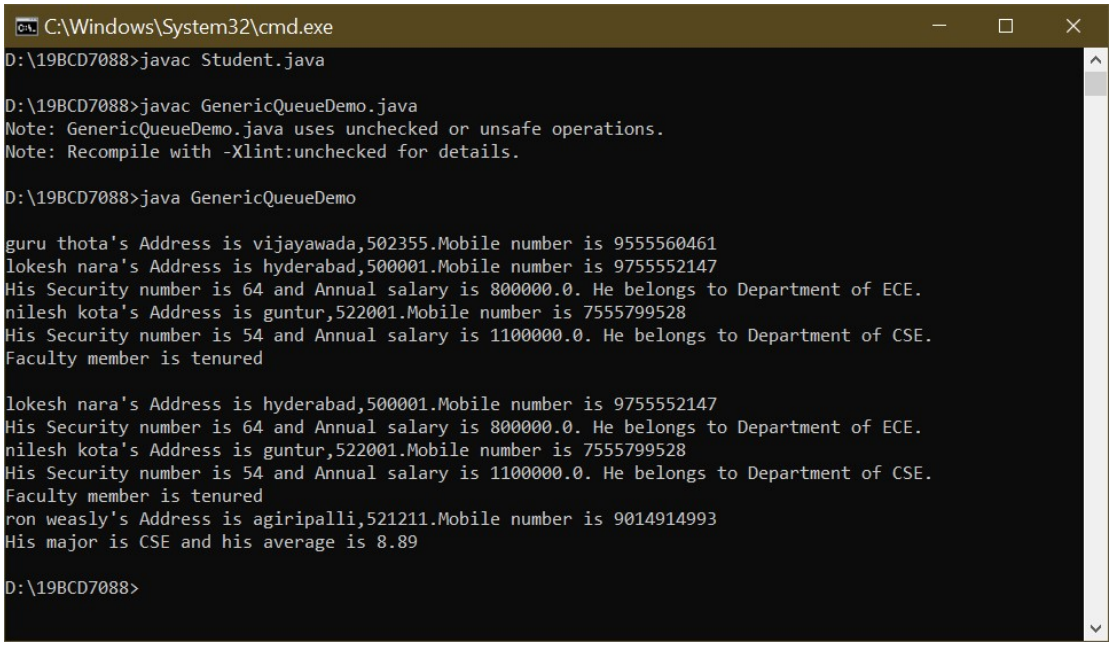
Here 'T' should be bounded by Person and its Children.(Refer Lab3 Exercise Question 5 to know the Person hierarchy).

Create GenericQueue<T> such that it implements GenericQueuable<T>. Write a GenericQueueDemo class to test the operations of GenericQueue class with two different queues.

```
interface GenericQueuable<T>{
	public void insertEnd(T e);
	public void removeBegin();
	public void printQueue();
	public boolean isQueueEmpty();
}
class GenericQueue<T> implements GenericQueuable<T>{
	int front,rear;
	int max;
	T a[];
	GenericQueue(int max){
		this.max=max;
		front=rear=0;
		a=(T[]) new Object[max];
	}
	public void insertEnd(T e){
		if(rear==max){
```

```
                System.out.println("Queue is full");
        }
        else{
                a[rear]=e;
                rear++;
        }
}
public void removeBegin(){
        if(front==rear){
                System.out.println("Queue is empty");
        }
        else{
                for(int j=0;j<rear-1;j++){
                   a[j]=a[j+1];
                }
                rear--;
        }
}
public void printQueue(){
        for(int i=front;i<rear;i++){
                System.out.print(a[i] + " ");
        }
        System.out.println();
}
public boolean isQueueEmpty(){
        if(rear==front){
                return true;
        }
        else{
                return false;
        }
```

```
	}
}
public class GenericQueueDemo{
	public static void main(String[] args) {
		GenericQueue<Person> g = new GenericQueue<Person>(3);
		Person p1=new Person("guru","thota","vijayawada",502355,9555560461L);
		Person p2=new
CollegeEmployee("lokesh","nara","hyderabad",500001,9755552147L,64,800000,"ECE");
		Person p3=new
Faculty("nilesh","kota","guntur",522001,7555799528L,54,1100000,"CSE",true);
		Person p4=new
Student("ron","weasly","agiripalli",521211,9014914993L,"CSE",8.89);
		g.insertEnd(p1);
		g.insertEnd(p2);
		g.insertEnd(p3);
		g.printQueue();
		g.removeBegin();
		g.insertEnd(p4);
		g.printQueue();
	}
}
```



2.a.

Create a generic Map interface MyMap<K,V> that represents a Map structure. K is the type for a key, V is the type of a value. A key is unique in a map and cannot be repeated.

The map contains the following abstract methods.

-an abstract method add(K key, V value) which adds the element to the map

-an abstract method remove(K key) which removes the element with the specified key from the map and returns the value removed.

-a abstract method size() which returns the size of the map

-a abstract method isEmpty() that would return true if the map is empty, and false otherwise.

-a abstract method keys() that returns the list of all keys.

-a abstract method print() that prints all the elements of the map.

2.b.

Create a generic class MyMapImpl that implements the interface MyMap. Use an ArrayList or array to store the keys, and another ArrayList or array to store the values.

2.c.

Create a test class "MyMapTest" that creates two Maps.

Map1: <String, Integer> where the key is a String and the value is an Integer

Map2: <Integer, Double> where the key is a Integer and the value is an Double

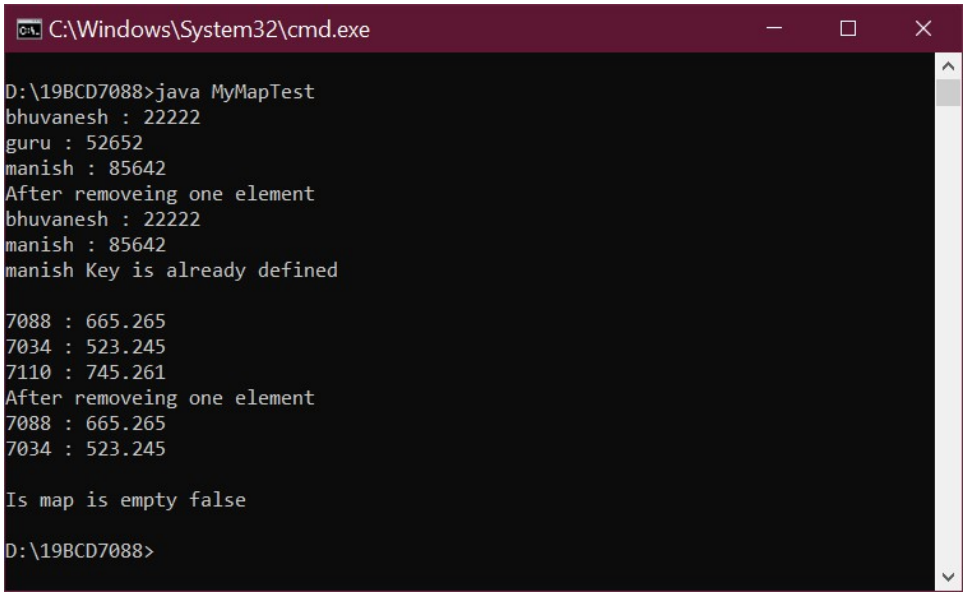
Add several elements. Try to add elements with the same key, and check that only one instance is added effectively with no redundancy.

```
interface MyMap<K,V>{
	public void add(K k,V v);
	public void remove(K k);
	public int size();
	public boolean isEmpty();
	public K []keys();
	public void print();
}
class MyMapImpl<K,V> implements MyMap<K,V>{
	K a[];
	V b[];
	int p,max,size,ind;
	boolean o;
	MyMapImpl(int n){
		p=-1;
		max=n;
		size = 0;
		a=(K[]) new Object[n];
		b=(V[]) new Object[n];
	}
	public boolean isPresent(K k){
		for (int i=0;i<size;i++){
			if(k.equals(a[i])){
				o=true;
			}
		}
		if(o){
			return false;
		}
```

```
        else{
                return true;
        }
}
public void add(K k ,V v){
        if(isPresent(k)){
        if(size==max){
                System.out.println("Map is full");
        }
        else{
                p++;
                a[p]=k;
                b[p]=v;
                size ++;
        }
}
else{
        System.out.println(k +" Key is already defined");
}
}
public void remove(K k){
        if(size!=0){
        for(int j=0;j<size;j++){
                if(k.equals(a[j])){
                        ind=j;
                }
        }
        for(int l=ind;l<size-ind;l++){
                a[l]=a[l+1];
                b[l]=b[l+1];
        }
```

```
                size--;
                p--;
        }
        else{
                System.out.println("Map is empty");
        }
        }
        public int size(){
                return size;
        }
        public boolean isEmpty(){
                if(size==0){
                        return true;
                }
                else{
                        return false;
                }
        }
        public K []keys(){
                return a;
        }
        public void print(){
                for(int m=0;m<size;m++){
                        System.out.println(a[m] + " : " + b[m]);
                }
        }
}
public class MyMapTest{
        public static void main(String[] args) {
                MyMapImpl<String, Integer> r = new MyMapImpl<String, Integer>(3);
                MyMapImpl<Integer, Double> u = new MyMapImpl<Integer, Double>(3);
```

```
                r.add("bhuvanesh",22222);
                r.add("guru",52652);
                r.add("manish",85642);
                r.print();
                System.out.println("After removeing one element");
                r.remove("guru");
                r.print();
                r.add("manish",1245);
                System.out.println();
                u.add(7088,665.265);
                u.add(7034,523.245);
                u.add(7110,745.261);
                u.print();
                System.out.println("After removeing one element");
                u.remove(7110);
                u.print();
                System.out.println();
                System.out.println("Is map is empty " + u.isEmpty());
        }
}
```



Lab 7

L1 slot

Valiveti manikanta bhuvanesh

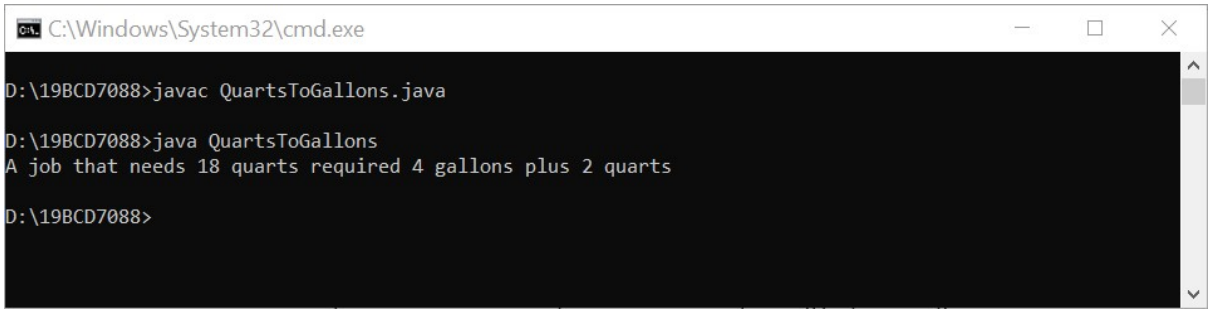
19BCD7088

1.

a.

Write a program that declares a named constant to hold the number of quarts in a gallon (4). Also declare a variable to represent the number of quarts needed for a painting job, and assign an appropriate value—for example, 18. Compute and display the number of gallons and quarts needed for the job. Display explanatory text with the values—for example, A job that needs 18 quarts requires 4 gallons plus 2 quarts. Save the program as QuartsToGallons.java.

```
public class  QuartsToGallons{

public static void main(String args[]) {

int qg = 4;

int n = 18;

System.out.println("A job that needs " + n + " quarts required " + n/qg + " gallons plus " +n%qg + " quarts");

}

}
```

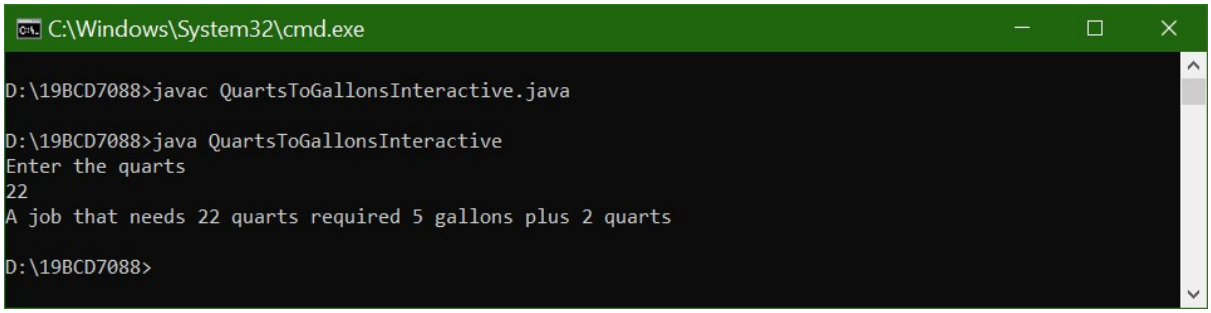


b.

Convert the QuartsToGallons program to an interactive application. Instead of assigning a value to the number of quarts, accept the value from the user as input. Save the revised program as QuartsToGallonsInteractive.java.

```
import java.util.Scanner;

public class  QuartsToGallonsInteractive{

public static void main(String args[]) {

        Scanner sc=new Scanner (System.in);

        int qg = 4;

        System.out.println("Enter the quarts");

        int n = sc.nextInt();
```

```
        System.out.println("A job that needs " + n + " quarts required " + n/qg + " gallons plus "
+n%qg + " quarts");

}

}
```

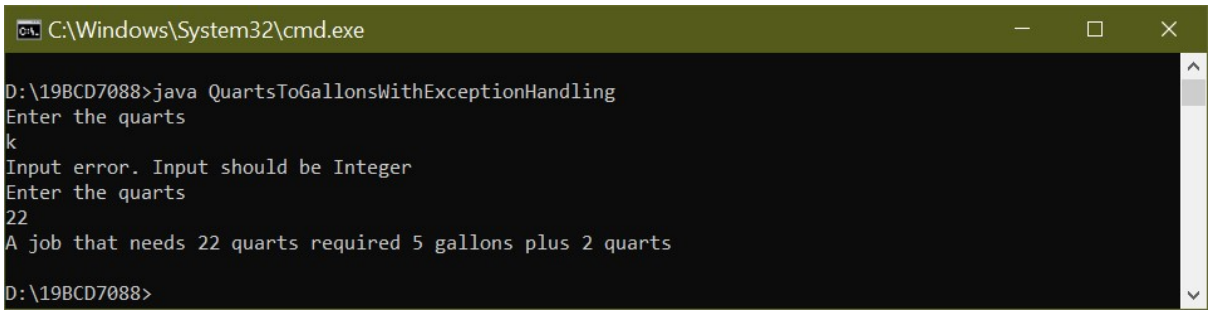


c.

Now, add exception-handling capabilities to this program and continuously reprompt the user while any nonnumeric value is entered. Save the file as QuartsToGallonsWithExceptionHandling.java.

```
import java.util.*;

public class QuartsToGallonsWithExceptionHandling{

        public static void main(String[] args) {

                Scanner sc=new Scanner (System.in);

                int qg = 4;

                int p=1;

                while(p!=0){

                try{

                        System.out.println("Enter the quarts");

                        int n = Integer.parseInt(sc.nextLine());

                        System.out.println("A job that needs " + n + " quarts required " + n/qg + "
gallons plus " +n%qg + " quarts");

                        p=0;

                }

                catch(Exception e){

                        System.out.println("Input error. Input should be Integer");

                }

        }

        }
```

}



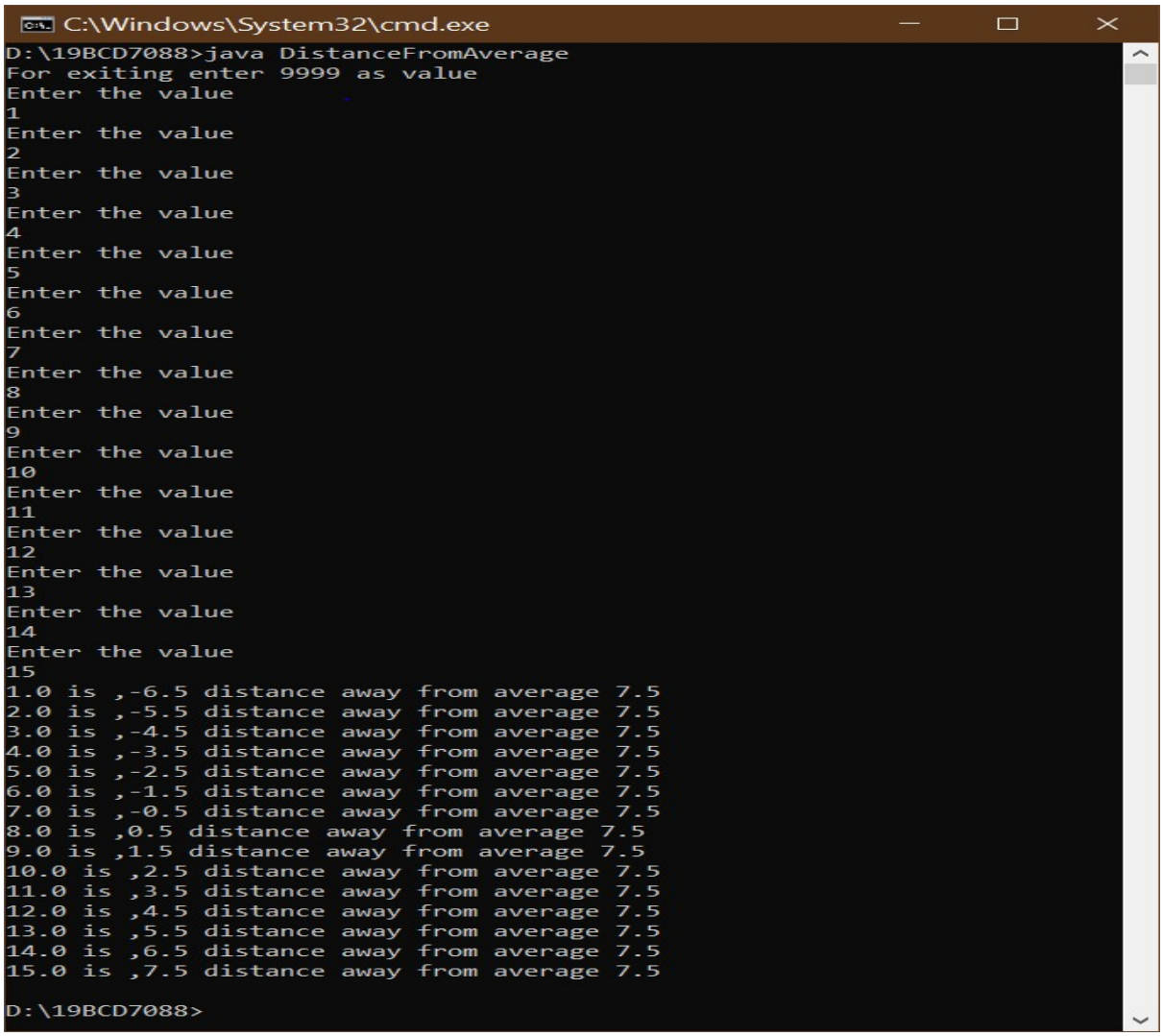
2.

a.

Allow a user to enter any number of double values up to 15. The user should enter 99999 to quit entering numbers. Display an error message if the user quits without entering any numbers; otherwise, display each entered value and its distance from the average. Save the file as DistanceFromAverage.java.

```
import java.util.Scanner;

public class DistanceFromAverage{

        public static void main(String[] args) {

                Scanner sc = new Scanner(System.in);

                double[] a=new double[15];

                double sum=0;

                int i=0,k=0;

                double avg,n=0;

                System.out.println("For exiting enter 9999 as value");

                while(i<a.length){

                        System.out.println("Enter the value");

                        a[i]=Double.parseDouble(sc.nextLine());

                        sum=sum+a[i];

                        n=a[i];

                        if(n==99999){

                           k=i;

                           i=a.length;

                        }

                        i++;

                }
```

```
                if(n==99999){
                   i=k;
                        System.out.println((a.length-i) + " values not entered");
                }
                else{
                        avg=sum/(i+1);
                        for(int j=0;j<a.length;j++){
                                System.out.println(a[j] + " is ," + (a[j]-avg) + " distance away from
average "+avg);
                        }
                }
        }
}
```



b.

Now, modify that program to first prompt the user to enter an integer that represents the array size. Java generates a NumberFormatException if you attempt to enter a noninteger value using nextInt(); handle this exception by displaying an appropriate error message. Create an array using the integer entered as the size.

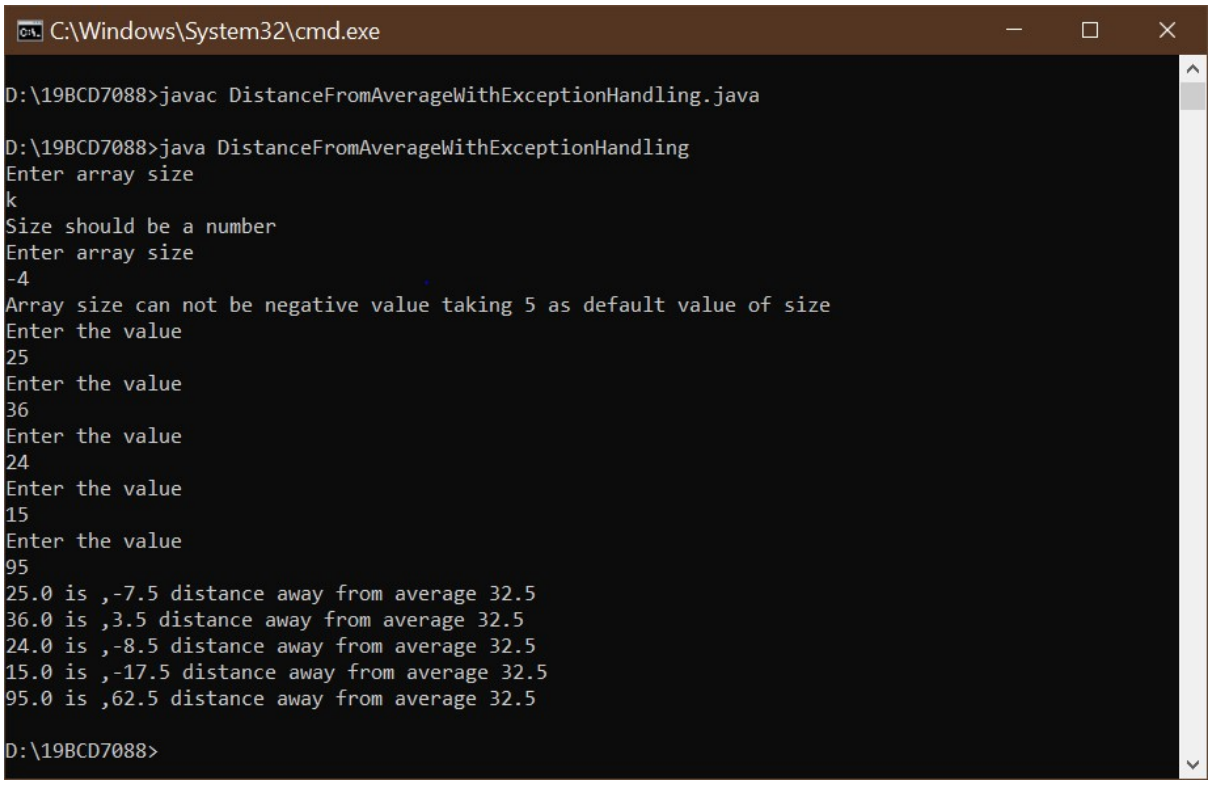
Java generates a NegativeArraySizeException if you attempt to create an array with a negative size; handle this exception by setting the array size to a default value of five.

If the array is created successfully, use exception-handling techniques to ensure that each entered array value is a double before the program calculates each element's distance from the average. Save the file as DistanceFromAverageWithExceptionHandling.java.

```
import java.util.*;

public class DistanceFromAverageWithExceptionHandling{

	public static void main(String[] args) {

		Scanner sc = new Scanner(System.in);

		double[] a=new double[5];

		double sum=0;

		int i=0,k=0;

		double avg,n=0;

		int size=5;

		int p=1;

		while(p!=0){

			try {

				System.out.println("Enter array size");

				size=Integer.parseInt(sc.nextLine());

				p=0;

				}

			catch (NumberFormatException e) {

				System.out.println("Size should be a number");

				}

		}

		try{

			a= new double[size];

		}
```

```
		catch(NegativeArraySizeException e){
			size=5;
			System.out.println("Array size can not be negative value taking 5 as default value of size");
		}
		while(i<a.length){
			try{
			System.out.println("Enter the value");
			a[i]=Double.parseDouble(sc.nextLine());
			sum=sum+a[i];
			n=a[i];
			if(n==9999){
			   k=i;
			   i=a.length;
			}
			i++;
		}
		catch(Exception e){
			System.out.println("Value should be double");
		}
		}
		if(n==9999){
		   i=k;
			System.out.println((a.length-i) + " values not entered");
		}
		else{
			avg=sum/(i+1);
			for(int j=0;j<a.length;j++){
				System.out.println(a[j] + " is ," + (a[j]-avg) + " distance away from average "+avg);
			}
		}
```

```
        }
}
```



3.

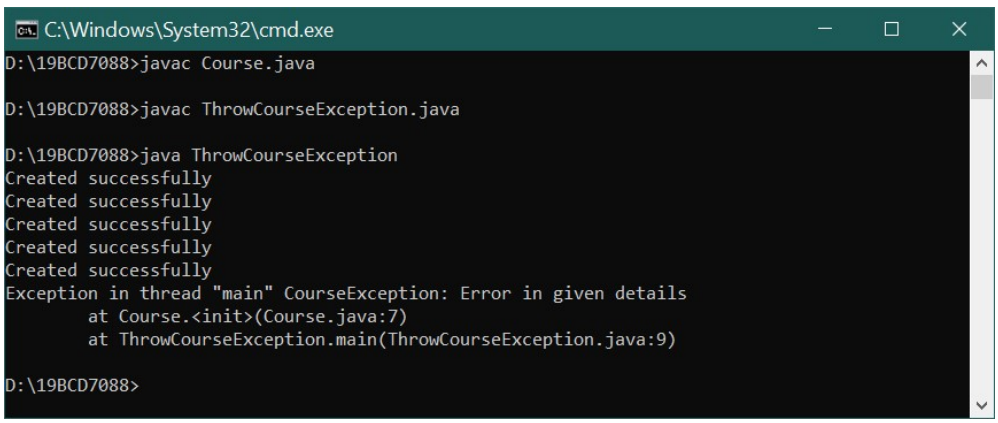
a.

Create a CourseException class that extends Exception and whose constructor receives a String that holds a college course's department (for example, CIS), a course number (for example, 101), and a number of credits (for example, 3). Save the file as CourseException.java. Create a Course class with the same fields and whose constructor requires values for each field. Upon construction, throw a CourseException if the department does not consist of three letters, if the course number does not consist of three digits between 100 and 499 inclusive, or if the credits are less than 0.5 or more than 6. Save the class as Course.java. Write an application that establishes an array of at least six Course objects with valid and invalid values. Display an appropriate message when a Course object is created successfully and when one is not. Save the file as ThrowCourseException.java.

```
class CourseException extends Exception{
    CourseException(String msg) {
                super(msg);
        }
}
```

```
class Course {
        String dept;
```

```
	int cn;
	double c;
	public Course(String dept, int cn, double c) throws CourseException {
		if(dept.length()!=3 || (cn<100 || cn>499) || (c<0.5 || c>6)) {
			throw new CourseException("Error in given details");
			}
			this.dept = dept;
			this.cn = cn;
			this.c = c;
			System.out.println("Created successfully");
		}
}
public class ThrowCourseException{
	public static void main(String[] args) throws CourseException{
		Course c[]=new Course[6];
		c[0]=new Course("BCE",350,5);
		c[1]=new Course("BCD",260,3);
		c[2]=new Course("BCI",275,4);
		c[3]=new Course("BCN",150,0.7);
		c[4]=new Course("BCR",185,1);
		c[5]=new Course("BCB",230,7);
	}
}
```



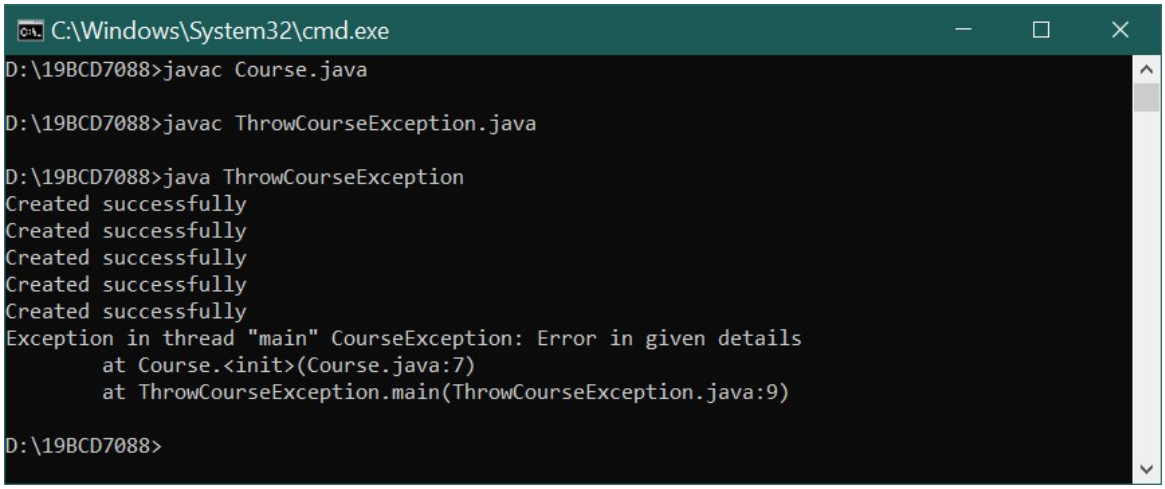
3.b.

Modify the CourseException class to extend RuntimeException class and identify the differences.

```
class CourseException extends RuntimeException{
    CourseException(String msg) {
                super(msg);
        }
}
```

```
class Course {
        String dept;
        int cn;
        double c;
        public Course(String dept, int cn, double c) throws CourseException {
                if(dept.length()!=3 || (cn<100 || cn>499) || (c<0.5 || c>6)) {
                        throw new CourseException("Error in given details");
                        }
                        this.dept = dept;
                        this.cn = cn;
                        this.c = c;
                        System.out.println("Created successfully");
                }
}
public class ThrowCourseException{
        public static void main(String[] args) throws CourseException{
                Course c[]=new Course[6];
                c[0]=new Course("BCE",350,5);
                c[1]=new Course("BCD",260,3);
                c[2]=new Course("BCI",275,4);
                c[3]=new Course("BCN",150,0.7);
                c[4]=new Course("BCR",185,1);
                c[5]=new Course("BCB",230,7);
        }
```

}


C:\Windows\System32\cmd.exe
D:\19BCD7088>javac Course.java
D:\19BCD7088>javac ThrowCourseException.java
D:\19BCD7088>java ThrowCourseException
Created successfully
Created successfully
Created successfully
Created successfully
Created successfully
Exception in thread "main" CourseException: Error in given details
at Course.<init>(Course.java:7)
at ThrowCourseException.main(ThrowCourseException.java:9)
D:\19BCD7088>

Lab 8

L1 slot

Valiveti manikanta bhuvanesh

19BCD7088

1.

Write a multithreaded program to compute sum of elements of NxN matrix. It should be done in two phases.

Phase I:

Create N threads to compute 'N' Columns sum, where 'i' th thread computes a 'i' th column sum.

Phase II:

Create a thread to compute sum of 'N' Column's Sums. Finally main thread is going to print result.

Note: Main thread should wait till all threads completes their work.

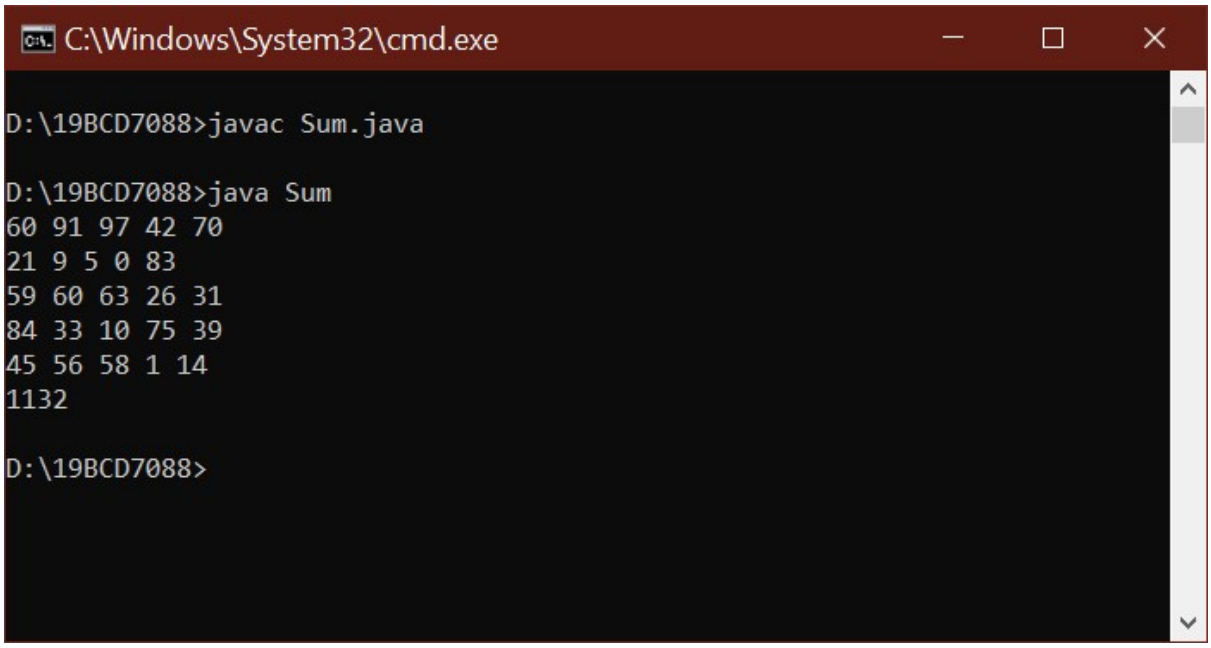
You can use Math.random() to initialize NxN matrix, in the range 0 to 100; 5<=N<=8.

Implement this application in two versions, using both Thread class & Runnable interface.

```
import java.lang.*;
import java.util.*;
public class Sum extends Thread{
                        public static int n=5;
                        public static  int sum;
                        public static int[][]a=new int[n][n];
                        public static int[]s=new int[n];
   static class ColSum extends Thread{
      int i;
      int sum=0;
      ColSum(int i){
         this.i=i;
      }
      public void run(){
         for(int q=0;q<n;q++){
            sum=sum+a[q][i];
         }
         s[i]=sum;
```

```
    }
}
static class Tsum extends Thread{
    public void run(){
                                    int sum1=0;
                                    for(int m=0;m<n;m++){
                                            sum1=sum1+s[m];
                }
                sum=sum1;
    }
}
        public static void main(String[] args) {
            Random r= new Random();
            for(int i=0;i<n;i++) {
                        for (int j=0;j<n;j++) {
                                a[i][j]=r.nextInt(100);
                                System.out.print(a[i][j] + " ");
                        }
                        System.out.println();
                }
                Sum.ColSum [] c=new Sum.ColSum[n];
                for(int k=0;k<n;k++){
                    c[k]=new Sum.ColSum(k);
                }
                Sum.Tsum t2=new Sum.Tsum();

        for(int y=0;y<n;y++){
            c[y].start();
        }
        t2.start();
        Thread t = Thread.currentThread();
```

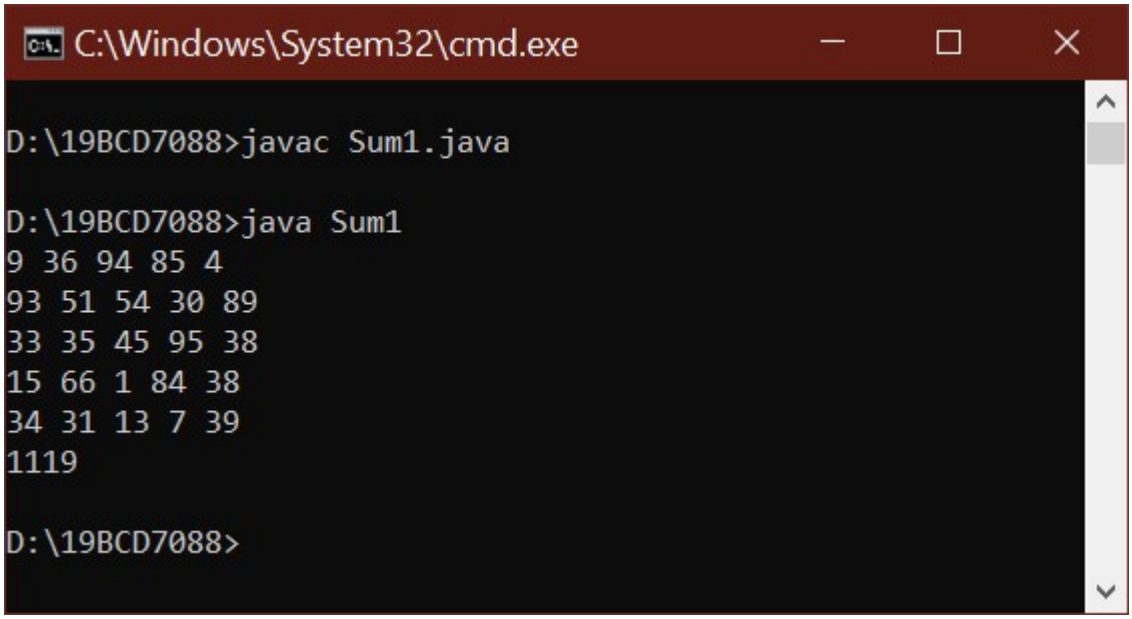
```
                    try{
                t.sleep(1500);
                }
                catch(Exception e){
                System.out.println(e);
                }
        System.out.println(sum);
}
}
```




```
import java.lang.*;
import java.util.*;
public abstract class Sum1 implements Runnable{
                public static int n=5;
                public static  int sum;
                public static int[][]a=new int[n][n];
                public static int[]s=new int[n];
   static class ColSum1 implements Runnable{
      int i;
      int sum=0;
```

```
    ColSum1(int i){
        this.i=i;
    }
    public void run(){
        for(int q=0;q<n;q++){
            sum=sum+a[q][i];
        }
        s[i]=sum;
    }
}
static class Tsum implements Runnable{
    public void run(){
                                    int sum1=0;
                                    for(int m=0;m<n;m++){
                                            sum1=sum1+s[m];
                }
                sum=sum1;
    }
}
        public static void main(String[] args) {
            Random r= new Random();
            for(int i=0;i<n;i++) {
                                for (int j=0;j<n;j++) {
                                        a[i][j]=r.nextInt(100);
                                        System.out.print(a[i][j] + " ");
                                }
                                System.out.println();
                }
                Thread [] t1=new Thread[n];
                for(int k=0;k<n;k++){
                    t1[k]=new Thread(new Sum1.ColSum1(k));
```

```
            }
            Thread t2=new Thread (new Sum1.Tsum());

        for(int y=0;y<n;y++){
           t1[y].start();
        }
        t2.start();
        Thread t = Thread.currentThread();
                        try{
              t.sleep(1500);
            }
            catch(Exception e){
               System.out.println(e);
            }
        System.out.println(sum);
}
}
```



2.

a.

Create a CeaserCipher class to perform substitution and reverse substitution of characters of a message.

- mEncryption method - substitute a character with another charcter of alphabet.

- mDecryption method - similar to mEncryption method but it perform in reverse.

Each character of message is considered as numeric value with the following mapping:a-z to 0-25, respectively.

(a-0,b-1,c-2,....z-25)

The mEncryption method replaces each character of the message with another character by using the following formula:(N(ch)+k)%26, where N(ch) means Numeric value of a character 'ch', k means key value 0<=k<=25.

The mDecryption method substitutes each character with the following formula: (N(ch)-k)%26.

Inputs to each method is a message and a key and output is substituted message printed on console character by character.

(Ex: Input to mEncryption is: rama and 25 and output is: qzlz ;

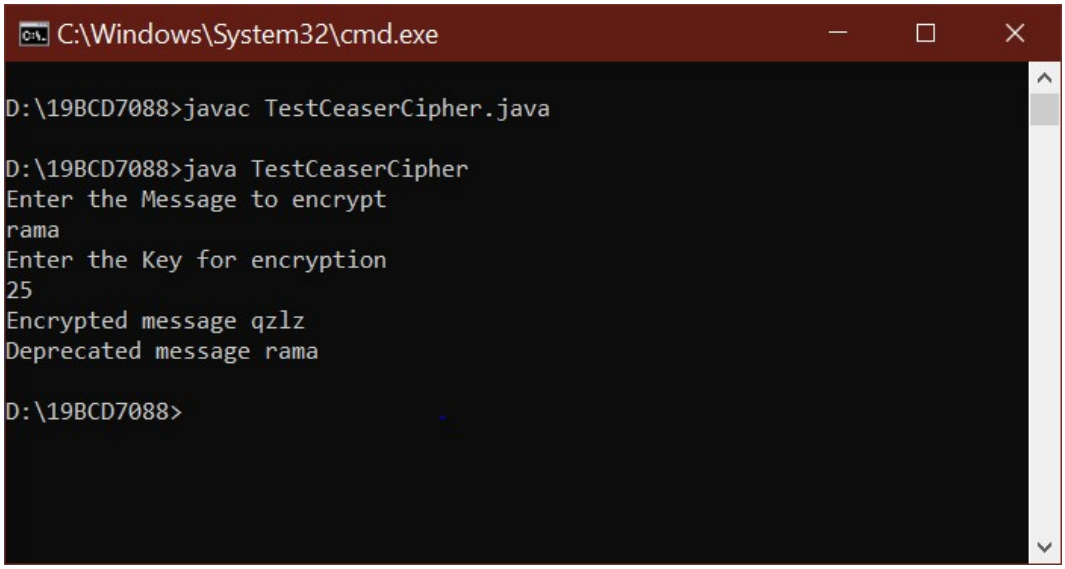
Input to mDecryption is: qzlz and 25 and output is: rama)

Create a TestCeaserCipher class to test mEncryption & mDecryption methods.

```
import java.util.Scanner;
class TestCeaserCipher
{
        public static String a = "abcdefghijklmnopqrstuvwxyz";
        public static String mEncryption(String p, int key)
        {
                p = p.toLowerCase();
    String c = "";
    for (int i = 0; i < p.length(); i++)
```

```
{
   int cp = a.indexOf(p.charAt(i));
   int kv = (key + cp) % 26;
   char rv = a.charAt(kv);
   c += rv;
}
return c;
     }
     public static String mDecryption(String c, int key)
     {
                c = c.toLowerCase();
String p = "";
for (int i = 0; i < c.length(); i++)
{
   int cp = a.indexOf(c.charAt(i));
   int kv = (cp - key) % 26;
   if (kv < 0)
   {
      kv = a.length() + kv;
   }
   char rv = a.charAt(kv);
   p += rv;
}
return p;
     }
     public static void main(String[] args)
     {
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter the Message to encrypt");
                String m =sc.nextLine();
                System.out.println("Enter the Key for encryption");
```

```
            int key = sc.nextInt();
            String c=mEncryption(m, key);
            System.out.println("Encrypted message " + c);
            System.out.println("Deprecated message " + mDecryption(c, key));
        }
}
```



2.

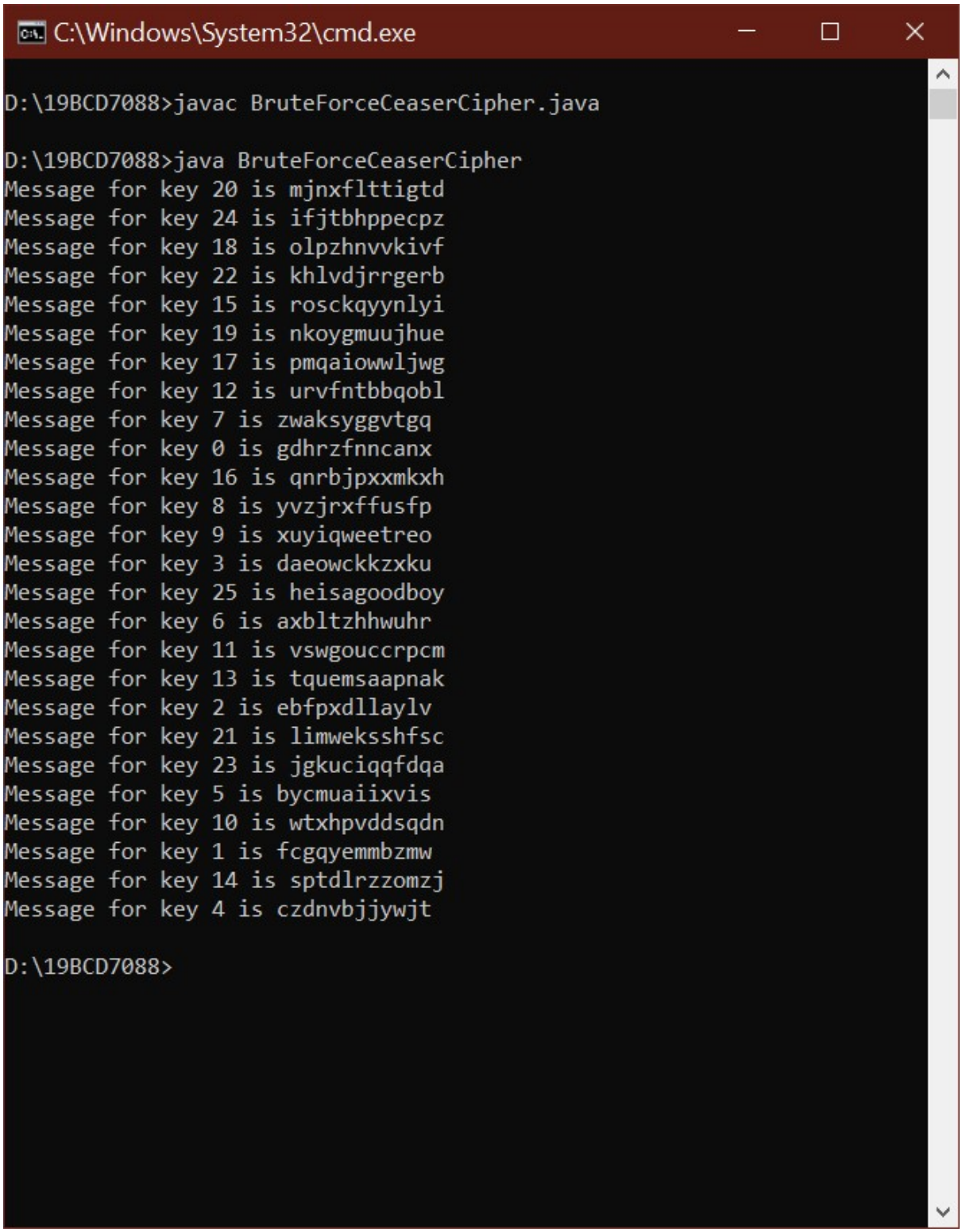
b.

Jennifer comes with a message "gdhrzfnncanx". She wants to perform reverse substitution using mDecryption method but not aware of key 'k'. To help her, develop a multithreaded program to create separate thread for each possible key 'k' and print all reverse substitutions. Do necessary changes to CeaserCipher class and provide synchronization for threads if the output from threads are mixed. Name the file as BruteForceCeaserCipher.java.

```
public class BruteForceCeaserCipher extends Thread
{
        public static String m = "gdhrzfnncanx";
        public static String a = "abcdefghijklmnopqrstuvwxyz";
        static class Decrypt extends Thread
        {
            int key;
             Decrypt(int key)
             {
```

```
                        this.key=key;
                }
                String l = m.toLowerCase();
        String p = "";
        public synchronized void decrypt(){
           for (int i = 0; i < l.length(); i++)
                                {
                                        int cp = a.indexOf(l.charAt(i));
                                        int kv = (cp - key) % 26;
                                        if (kv < 0)
                                        {
                                                kv = a.length() + kv;
                                        }
                                        char r = a.charAt(kv);
                                        p += r;
                                }
                                System.out.println("Message for key "+key+ " is " + p);
        }
                  public void run(){
                                  decrypt();
                }
}
          public static void main(String[] args)
          {
                  BruteForceCeaserCipher.Decrypt [] c=new BruteForceCeaserCipher.Decrypt[26];
                  for(int k=0;k<26;k++){
                     c[k]=new BruteForceCeaserCipher.Decrypt(k);
                  }
                  for(int y=0;y<26;y++){
             c[y].start();
          }
```

```
}


}
```


C:\Windows\System32\cmd.exe
D:\19BCD7088>javac BruteForceCeaserCipher.java
D:\19BCD7088>java BruteForceCeaserCipher
Message for key 20 is mjnxflttigtd
Message for key 24 is ifjtbhppecpz
Message for key 18 is olpzhnvvkivf
Message for key 22 is khlvdjrrgerb
Message for key 15 is rosckqyynlyi
Message for key 19 is nkoygmuujhue
Message for key 17 is pmqaiowwljwg
Message for key 12 is urvfntbbqobl
Message for key 7 is zwaksyggvtgq
Message for key 0 is gdhrzfnncanx
Message for key 16 is qnrbjpxxmkxh
Message for key 8 is yvzjrxffusfp
Message for key 9 is xuyiqweetreo
Message for key 3 is daeowckkzxku
Message for key 25 is heisagoodboy
Message for key 6 is axbltzhhwuhr
Message for key 11 is vswgouccrpcm
Message for key 13 is tquemsaapnak
Message for key 2 is ebfpxdllaylv
Message for key 21 is limweksshfsc
Message for key 23 is jgkuciqqfdqa
Message for key 5 is bycmuaiixvis
Message for key 10 is wtxhpvddsqdn
Message for key 1 is fcgqyemmbzmw
Message for key 14 is sptdlrzzomzj
Message for key 4 is czdnvbjjywjt
D:\19BCD7088>

Lab 9

L1 slot

Valiveti manikanta bhuvanesh

19BCD7088

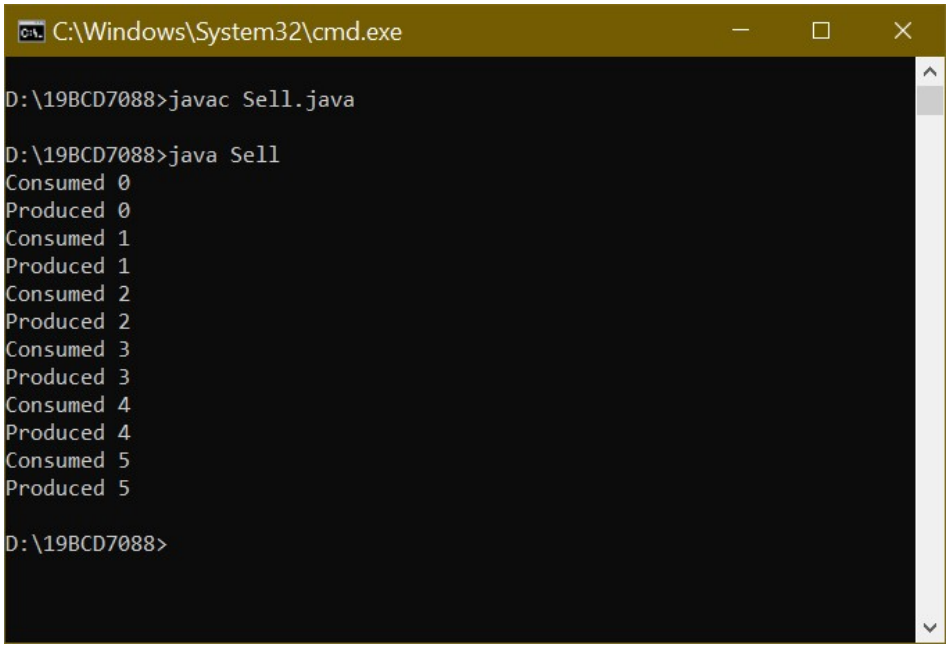
1. Apply Interthread communication to solve the Producer-Consumer problem with a common or shared bounded buffer(Queue) holding upto 5 elements. The producer consumer problem is a synchronization problem. There is a fixed size buffer and the producer produces items and enters them into the buffer. The consumer removes the items from the buffer and consumes them. A producer should not produce items into the buffer when the consumer is consuming an item from the buffer and vice versa. So the buffer should only be accessed by the producer or consumer at a time. When ever buffer is filled up and no more space to add the element into the queue(buffer) producerhas to wait until the buffer is emptied by consumer. When ever the buffer is empty and no more items are available for consumption the consumer should wait for producer to produceelements. Write a solution for N elements, where N is multiple of 5 or greater than 5 other than0

```
import java.util.*;

class Queue

{

int n;

boolean run = false;

synchronized int get()

{

while(!run)

try

{

wait();

}

catch(InterruptedException e)

{

System.out.println(e);

}

System.out.println("Produced " + n);

run=false;

notify();
```

```
return n;
}
synchronized void set(int n)
{
while(run)
try
{
wait();
}
catch(InterruptedException e)
{
System.out.println(e);
}
this.n = n;
run = true;
System.out.println("Consumed " + n);
notify();
}
}
class Producer implements Runnable
{
Queue q;
Thread t;
Producer(Queue q)
{
this.q=q;
t=new Thread(this, "Producer");
}
public void run()
{
int i = 0;
```

```
while (true)
{
if(i>5)
{
System.exit(0);
}
q.set(i++);
}
}
}
class Consumer implements Runnable
{
Queue q;
Thread t;
Consumer (Queue q)
{
this.q = q;
t=new Thread (this,"Consumer");
}
public void run ()
{
while(true)
{
q.get();
}
}
}
class Sell
{
public static void main(String args[])
{
```

```
Queue q =new Queue();

Producer p = new Producer (q);

Consumer c = new Consumer (q);

p.t.start();

c.t.start();

}

}
```



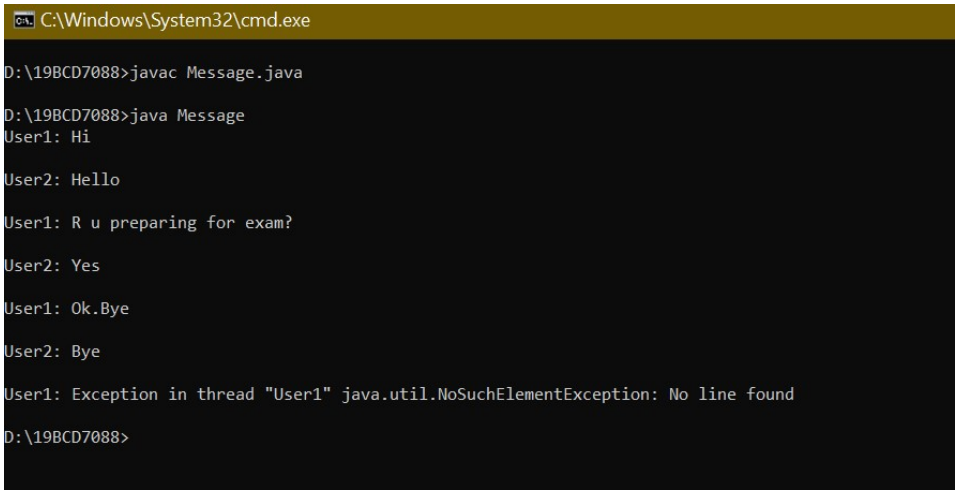
2. Develop a simple chat application between two users using "Send-Wait-Receive" Protocol: Once a user sends a message, he waits till a message is received from other user. The users are "User1" and "User2". At initial stage of application, User1 is in sending mode and User2 is in receiving mode. These two users are sending and receiving the messages alternatively. -Create a Chat class with two methods: sendMessage and recvMessage – Create two threads to represent two users,User1 and User2. -Use Interthread communication to exchange messages. -No need to maintain any chat history. Example: User1: Hi User2: Hello User1: R u preparing for exam? User2: Yes User1: Ok. Bye User2: Bye.

```
import java.util.*;

class Queue1

{

Scanner sc=new Scanner(System.in);

int n;

String msg;
```

```
boolean run = false;
synchronized int recvMessage()
{
while(!run)
 try
{
wait();
}
catch(InterruptedException e)
{
System.out.println(e);
}
System.out.print("User2: ");
msg=sc.nextLine();
run = true;
System.out.println();
run=false;
notify();
return n;
}
synchronized void sendMessage(int n)
{
while(run)
            try
{
wait();
}
catch(InterruptedException e)
{
System.out.println(e);
}
```

```
this.n = n;
System.out.print("User1: ");
msg=sc.nextLine();
run = true;
System.out.println();
notify();
}
}
class User1 implements Runnable
{
Queue1 que;
Thread t;
User1(Queue1 que)
{
this.que=que;
t=new Thread(this, "User1");
}
public void run()
{
int i = 0;
while (true)
{
if(i>5)
{
System.exit(0);
}
que.sendMessage(i++);
}
}
}
class User2 implements Runnable
```

```
{
Queue1 que;
Thread t;
User2 (Queue1 que)
{
this.que = que;
t=new Thread (this,"User2");
}
public void run ()
{
while(true)
{
que.recvMessage();
}
}
}
public class Message{
public static void main(String args[])
{
Queue1 que =new Queue1();
User1 u1 = new User1 (que);
User2 u2 = new User2 (que);
u1.t.start();
u2.t.start();
}
}
```


C:\Windows\System32\cmd.exe
D:\19BCD7088>javac Message.java
D:\19BCD7088>java Message
User1: Hi
User2: Hello
User1: R u preparing for exam?
User2: Yes
User1: Ok.Bye
User2: Bye
User1: Exception in thread "User1" java.util.NoSuchElementException: No line found
D:\19BCD7088>

Lab 10

L1 slot

1.

Valiveti manikanta bhuvanesh

19BCD7088

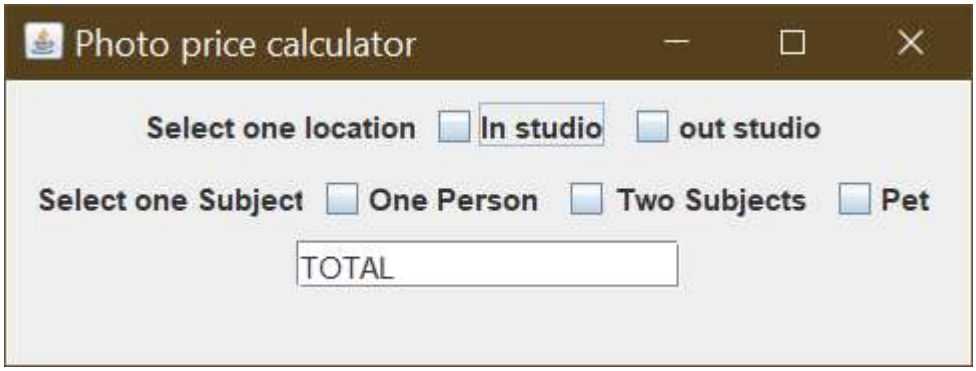
Create an application for Paula's Portraits, a photography studio. The application allows users to compute the price of a photography session. Paula's base price is $40 for an in-studio photo session with one person. The in-studio fee is $75 for a session with two or more subjects, and $95 for a session with a pet. A $90 fee is added to take photos on location instead of in the studio. Include a set of mutually exclusive check boxes to select the portrait subject and another set for the session location. Include labels as appropriate to explain the application's functionality. Save the file as JPhotoFrame.java.

```
import java.awt.*;

import java.awt.event.*;

import javax.swing.*;

class MyWindowAdapter extends WindowAdapter{

  public void windowClosing(WindowEvent we) {

    System.exit(0);

  }


}

class DataType{

        int single=40;

        int dual=75;

        int pet=95;

        int onl=90;

        int sum=0;

        JCheckBox ib,ob,sb,tb,pb;

        JTextField total;

}

class CheckMeListener extends DataType implements ItemListener{

        public void itemStateChanged(ItemEvent e){

                Object source=e.getSource();

                int select=e.getStateChange();

                if(source==ib)

                {
```

```
if(select==ItemEvent.SELECTED){
sum+=single;
}
else{
sum-=single;
}
}
else if(source==ob)
{

if(select==ItemEvent.SELECTED){
sum+=onl;
}
else{
sum-=onl;
}
}
else if(source==sb)
{
if(select==ItemEvent.SELECTED){
sum+=single;
}
else{
sum-=single;
}
}
else if(source==tb)
{
if(select==ItemEvent.SELECTED){
sum+=dual;
}
```

```
                    else{
                    sum-=dual;
                    }
                    }

                    else if(source==pb)
                    {
                    if(select==ItemEvent.SELECTED){
                    sum+=pet;
                    }
                    else{
                    sum-=pet;
                    }
                    }
                    total.setText("$"+sum);
                            }
}
public class JPhotoFrame{
        public static void main(String[] args) {
                    CheckMeListener y=new CheckMeListener();
                    JFrame f = new JFrame("Photo price calculator");
    f.setSize(400,150);
    Panel p = new Panel();
    p.setLayout(new FlowLayout());
    y.ib=new JCheckBox("In studio");
    y.ob=new JCheckBox("out studio");
    y.sb=new JCheckBox("One Person");
                    y.tb=new JCheckBox("Two Subjects");
                    y.pb=new JCheckBox("Pet");
                    y.total=new JTextField("TOTAL",15);
    ButtonGroup lg=new ButtonGroup();
```

```
        lg.add(y.ib);
        lg.add(y.ob);
        p.add(new JLabel("Select one location"));
        p.add(y.ib);
        p.add(y.ob);
        ButtonGroup m=new ButtonGroup();
                        m.add(y.sb);
                        m.add(y.tb);
                        m.add(y.pb);
                        p.add(new JLabel("Select one Subject"));
                        p.add(y.sb);
                        p.add(y.tb);
                        p.add(y.pb);
                        y.ib.addItemListener(y);
                        y.ob.addItemListener(y);
                        y.sb.addItemListener(y);
                        y.tb.addItemListener(y);
                        y.pb.addItemListener(y);
                        p.add(y.total);
                        f.add(p);
                f.addWindowListener(new MyWindowAdapter());
                f.setVisible(true);
                                }
}
```



2.

Write an application for Lambert's Vacation Rentals. Use separate ButtonGroups to allow a client to select one of three locations, the number of bedrooms, and whether meals are included in the rental. Assume that the locations are parkside for $600 per week, poolside for $750 per week, or lakeside for $825 per week. Assume that the rentals have one, two, or three bedrooms and that each bedroom greater than one adds $75 to the base price. Assume that if meals are added, the price is $200 more per rental. Save the file as JVacationRental.java.

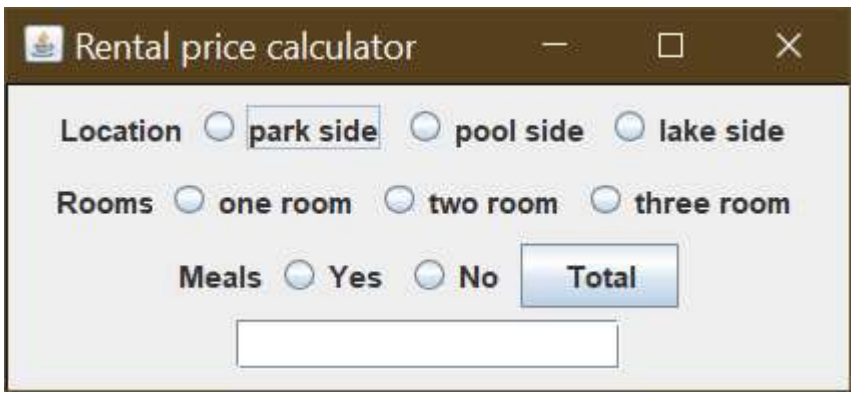
```
import java.awt.*;

import java.awt.event.*;

import javax.swing.*;

class MyWindowAdapter extends WindowAdapter{

  public void windowClosing(WindowEvent we) {

    System.exit(0);

  }


}


class Listeners {

   static int pars= 600;

   static int pos = 750;

   static int ls = 825;

   static int b1= 75;

   static int b2 = 150;

   static int b3 = 225;

   static int m = 200;

   static int lr = 0;

   static int br = 0;

   static int mc = 0;

   static JRadioButton park,pool,lake,one,two,three,yes,no;

   static JButton Cal;

   static JTextField total;

  static class ClickMeListener implements ActionListener{
```

```
  public void actionPerformed(ActionEvent e)

  {
   Object source = e.getSource();
  if(source == Cal){
   double totalRent = lr + br + mc;
   total.setText("Total amount $ " + totalRent);
  }
 }
}
 static class blistener implements ItemListener{
  public void itemStateChanged(ItemEvent e){
    Object source = e.getItem();
    if(source == one){
     br = b1;
    }
    else if(source == two){
     br = b2;
    }
    else if(source == lake){
     br=b3;
    }
    else{
     br=0;
    }
  }
}
static class llistener implements ItemListener{
 public void itemStateChanged(ItemEvent e)
 {
   Object source = e.getItem();
```

```
    if(source == park){
      lr = pars;
    }
    else if(source == pool){
      lr = pos;
    }
    else if(source == lake){
      lr = ls;
    }
    else{
      lr=0;
    }
  }
}
static class mlistener implements ItemListener{
  public void itemStateChanged(ItemEvent e)
  {
    Object source = e.getItem();
    if(source == yes){
       mc = m;
    }
    else if(source == no){
      mc = 0;
    }
    else{
      mc= 0;
    }
    }
}
}
public class JVacationRental{
```

```
public static void main(String[] args) {
  Listeners y = new Listeners();
  Frame f = new JFrame("Rental price calculator");
  f.setSize(350,160);
  Panel p = new Panel();
  p.setLayout(new FlowLayout());
  y.park= new JRadioButton("park side");
  y.pool= new JRadioButton("pool side");
  y.lake= new JRadioButton("lake side");
  y.one= new JRadioButton("one room");
  y.two= new JRadioButton("two room");
  y.three= new JRadioButton("three room");
  y.yes= new JRadioButton("Yes");
  y.no= new JRadioButton("No");
  y.Cal= new JButton("Total");
  y.total=new JTextField(15);
  ButtonGroup l=new ButtonGroup();
  l.add(y.park);
  l.add(y.pool);
  l.add(y.lake);
  ButtonGroup b=new ButtonGroup();
  b.add(y.one);
  b.add(y.two);
  b.add(y.three);
  ButtonGroup yn=new ButtonGroup();
  yn.add(y.yes);
  yn.add(y.no);
  p.add(new JLabel("Location"));
  p.add(y.park);
  p.add(y.pool);
  p.add(y.lake);
```

```
    p.add(new JLabel("Rooms"));
    p.add(y.one);
    p.add(y.two);
    p.add(y.three);
    p.add(new JLabel("Meals"));
    p.add(y.yes);
    p.add(y.no);
    p.add(y.Cal);
    p.add(y.total);
    y.park.addItemListener(new Listeners.llistener());
    y.pool.addItemListener(new Listeners.llistener());
    y.lake.addItemListener(new Listeners.llistener());
    y.one.addItemListener(new Listeners.blistener());
    y.two.addItemListener(new Listeners.blistener());
    y.three.addItemListener(new Listeners.blistener());
    y.yes.addItemListener(new Listeners.mlistener());
    y.no.addItemListener(new Listeners.mlistener());
    y.Cal.addActionListener(new Listeners.ClickMeListener());
    f.add(p);
    f.addWindowListener(new MyWindowAdapter());
    f.setVisible(true);
  }
}
```



Lab 11

L1 slot

Valiveti manikanta bhuvanesh

19BCD7088

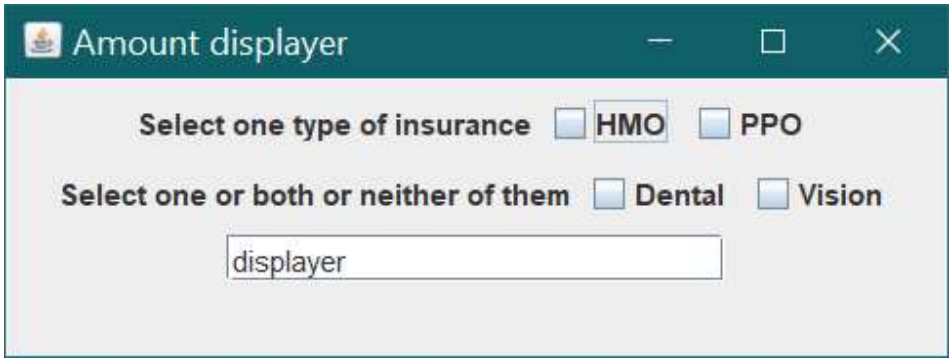
1.

Write a JavaFX application that allows the user to choose insurance options. Use a ToggleGroup to allow the user to select only one of two 8 types—HMO (health maintenance organization) or PPO (preferred provider organization). Use CheckBoxes for dental insurance and vision insurance options; the user can select one option, both options, or neither option. As the user selects each option, display its name and price in a text field; the HMO costs $200 per month, the PPO costs $600 per month, the dental coverage adds $75 per month, and the vision care adds $20 per month. Save the application as FXInsurance.java.

```
import java.awt.*;

import java.awt.event.*;

import javax.swing.*;

class MyWindowAdapter extends WindowAdapter{

 public void windowClosing(WindowEvent we) {

  System.exit(0);

 }


}

class CheckMeListener implements ItemListener {

        JCheckBox hb,pb,db,vb;

        JTextField disp;

        public void itemStateChanged(ItemEvent e){

                Object source=e.getSource();

                int select=e.getStateChange();

                if(source==hb){

                        if(select==ItemEvent.SELECTED){

                                disp.setText("HMO costs $200 per month");

                        }

                        else{

                                disp.setText("");

                                }

                }

                else if(source==pb){
```

```
                    if(select==ItemEvent.SELECTED){
                        disp.setText("PPO costs $600 per month");
                    }
                    else{
                        disp.setText("");
                    }
                }
                else if(source==db){
                    if(select==ItemEvent.SELECTED){
                        disp.setText("dental coverage adds $75 per month");
                    }
                    else{
                        disp.setText("");
                    }
                }
                else if(source== vb){
                    if(select==ItemEvent.SELECTED){
                        disp.setText("vision care adds $20 per month");
                    }
                    else{
                        disp.setText("");
                    }
                }
                else{
                    disp.setText("");
                }

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

```
                CheckMeListener y=new CheckMeListener();
                JFrame f = new JFrame("Amount displayer");
        f.setSize(400,150);
        Panel p = new Panel();
        p.setLayout(new FlowLayout());
        y.hb=new JCheckBox("HMO");
        y.pb=new JCheckBox("PPO");
        y.db=new JCheckBox("Dental");
        y.vb=new JCheckBox("Vision");
        y.disp=new JTextField("displayer",20);
        ButtonGroup bg=new ButtonGroup();
                bg.add(y.hb);
                bg.add(y.pb);
                p.add(new JLabel("Select one type of insurance"));
                p.add(y.hb);
                p.add(y.pb);
                p.add(new JLabel("Select one or both or neither of them"));
                p.add(y.db);
                p.add(y.vb);
                p.add(y.disp);
                y.hb.addItemListener(y);
                y.pb.addItemListener(y);
                y.db.addItemListener(y);
                y.vb.addItemListener(y);
                f.add(p);
            f.addWindowListener(new MyWindowAdapter());
            f.setVisible(true);
          }
}
```



2.

Design a JavaFX application for the Sublime Sandwich Shop. The user makes sandwich order choices from list boxes, and the application displays the price. The user can choose from three main sandwich ingredients of your choice (for example, chicken) at three different prices. The user also can choose from three different bread types (for example, rye) from a list of at least three options.

Save the application as FXSandwich.java.

```
import java.awt.*;

import java.awt.event.*;

import javax.swing.*;

import javax.swing.event.*;

class MyWindowAdapter extends WindowAdapter{

  public void windowClosing(WindowEvent we) {

    System.exit(0);

  }


}

class Listener implements ListSelectionListener {

            int ing,bread;

            JList ingl,breads;

            JTextField total;

             public void valueChanged(ListSelectionEvent e){

                        int indexing = ingl.getSelectedIndex();

                        int indexbread = breads.getSelectedIndex();

                        if(indexing==0){

                                    ing=200;
```

```
                }
                else if(indexing==1){
                        ing=180;
                }
                else if(indexing==2){
                        ing=118;
                }
                else{
                        ing=0;
                }
                if(indexbread==0){
                        bread=30;
                }
                else if(indexbread==1){
                        bread=50;
                }
                else if(indexbread==2){
                        bread=70;
                }
                else{
                        bread=0;
                }
                double sum= ing+bread;
                total.setText("Total price $"+sum);
        }

}
public class FXSandwich{
        public static void main(String[] args) {
        String Ingredients [] = {"Panner", "Mushroom", "Potato"};
        String breadTypes[] = {"Wheat", "Rye", "Brioche"};
```

```
        Listener y = new Listener();
    Frame f = new JFrame("Sublime Sandwich Shop");
    f.setSize(350,160);
    Panel p = new Panel();
    p.setLayout(new FlowLayout());
    y.total=new JTextField(15);
    y.ingl = new JList<String>(Ingredients);
    y.breads = new JList<String>(breadTypes);
    p.add(new JLabel("Choose Ingredients"));
    p.add(y.ingl);
    p.add(new JLabel("Choose Bread"));
    p.add(y.breads);
    p.add(y.total);
    y.ingl.addListSelectionListener(y);
    y.breads.addListSelectionListener(y);
    f.add(p);
    f.addWindowListener(new MyWindowAdapter());
    f.setVisible(true);
        }
}
```

Lab12

L1 slot

Valiveti manikanta bhuvanesh

19BCD7088

Develop a Menu Based GUI using Swings:

Menus:

-Country

-Font

-Color

Menu Items:

-Any 7 country names under Country.

-Name,Type,Size are Menu items under Font.

---Any four font names under Name Menu

---Type should contain Bold,Italic

---Size may contain 12,14,16,18

-Any standard seven colors as Menu items under Color Menu.

If a user clicks any menu item, It should be displayed with specified color and font in window area.

Provide shortcut keys to every menus and menu items like ctrl+o, ctrl+s.

Provide a popup menu to contain two options, clear and exit. When user clicks clear, it should clears the window area and when user clicks exit, it should terminate the application.

Save the file as MenuWithSwings.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
public class MenuWithSwing implements ActionListener{
        static JFrame f;
        static Panel p;
        static JMenuBar mb;
```

```
	static JMenu country,font,name,type,size,colour;
	static JMenuItem
times,dia,sansserif,courier,bold,italic,size12,size14,size16,size18,india,aus,brazil,germ,ice,mal,rus,blu
,bro,gray,green,org,red,yel,clc,exit;
	static JLabel disp;
	static JPopupMenu pop;
	public static void main(String[] args) {
		MenuWithSwing m = new MenuWithSwing();
		f = new JFrame("MenuWithSwings ");
	f.setSize(150,250);
	Panel p = new Panel();
	p.setLayout(new FlowLayout());
	pop =new JPopupMenu();
	clc=new JMenuItem("Clear");
	clc.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_C, KeyEvent.CTRL_MASK));
	exit=new JMenuItem("Exit");
	exit.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_E, KeyEvent.CTRL_MASK));
	pop.add(clc);
	clc.addActionListener(m);
	pop.add(exit);
	exit.addActionListener(m);
	mb = new JMenuBar();
	country =new JMenu("Country");
	font = new JMenu("Font");
	colour = new JMenu("Colour");
	name = new JMenu("Name");
	type = new JMenu("Type");
	size = new JMenu("Size");
	country.setMnemonic('C');
	font.setMnemonic('F');
	colour.setMnemonic('L');
	name.setMnemonic('N');
```

```
type.setMnemonic('T');
size.setMnemonic('S');
times=new JMenuItem("TimesRoman");
dia=new JMenuItem("Dialog");
sansserif = new JMenuItem("SansSerif");
courier=new JMenuItem("Courier");
bold = new JMenuItem("Bold");
italic = new JMenuItem("Italic");
size12 = new JMenuItem("Size 12");
size14 = new JMenuItem("Size 14");
size16 = new JMenuItem("Size 16");
size18 = new JMenuItem("Size 18");
india = new JMenuItem("India");
aus =new JMenuItem("Australia");
brazil = new JMenuItem("Brazil");
germ = new JMenuItem("Germany");
ice = new JMenuItem("Iceland");
mal = new JMenuItem("Malaysia");
rus = new JMenuItem("Russia");
blu = new JMenuItem("Blue");
bro =new JMenuItem("Brown");
gray = new JMenuItem("Gray");
green = new JMenuItem("Green");
org = new JMenuItem("Orange");
red = new JMenuItem("Red");
yel=new JMenuItem("Yellow");
times.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_I, KeyEvent.CTRL_MASK));
dia.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_D, KeyEvent.CTRL_MASK));
sansserif.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_A, KeyEvent.CTRL_MASK));
courier.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_U, KeyEvent.CTRL_MASK));
bold.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_B, KeyEvent.CTRL_MASK));
```

```
italic.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_R, KeyEvent.CTRL_MASK));
size12.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F2, 0));
size14.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F4, 0));
size16.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F6, 0));
size18.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F8, 0));
india.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_G, KeyEvent.CTRL_MASK));
aus.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_H, KeyEvent.CTRL_MASK));
brazil.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_J, KeyEvent.CTRL_MASK));
germ.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_K, KeyEvent.CTRL_MASK));
ice.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_M, KeyEvent.CTRL_MASK));
mal.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_P, KeyEvent.CTRL_MASK));
rus.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_Q, KeyEvent.CTRL_MASK));
blu.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_V, KeyEvent.CTRL_MASK));
bro.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_W, KeyEvent.CTRL_MASK));
gray.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_X, KeyEvent.CTRL_MASK));
green.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_Y, KeyEvent.CTRL_MASK));
org.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_Z, KeyEvent.CTRL_MASK));
red.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F1, 0));
yel.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F3, 0));
disp = new JLabel();
country.add(india);
country.add(aus);
country.add(brazil);
country.add(germ);
country.add(ice);
country.add(mal);
country.add(rus);
mb.add(country);
name.add(times);
name.add(dia);
name.add(sansserif);
```

```
name.add(courier);
font.add(name);
type.add(bold);
type.add(italic);
font.add(type);
size.add(size12);
size.add(size14);
size.add(size16);
size.add(size18);
font.add(size);
mb.add(font);
colour.add(blu);
colour.add(bro);
colour.add(gray);
colour.add(green);
colour.add(org);
colour.add(red);
colour.add(yel);
mb.add(colour);
india.addActionListener(m);
aus.addActionListener(m);
brazil.addActionListener(m);
germ.addActionListener(m);
ice.addActionListener(m);
mal.addActionListener(m);
rus.addActionListener(m);
times.addActionListener(m);
dia.addActionListener(m);
sansserif.addActionListener(m);
courier.addActionListener(m);
bold.addActionListener(m);
```

```
italic.addActionListener(m);
size12.addActionListener(m);
size14.addActionListener(m);
size16.addActionListener(m);
size18.addActionListener(m);
blu.addActionListener(m);
bro.addActionListener(m);
gray.addActionListener(m);
green.addActionListener(m);
org.addActionListener(m);
red.addActionListener(m);
yel.addActionListener(m);
p.addMouseListener(new MouseAdapter() {
   public void mouseClicked(MouseEvent e) {
      pop.show(p , e.getX(), e.getY());
   }
 });
p.add(mb);
p.add(disp);
p.add(pop);
f.add(p);
f.setVisible(true);
     }
     public void actionPerformed(ActionEvent o) {
                if(o.getSource()==clc){
                        disp.setText("");
                }
                if(o.getSource()==exit){
                        System.exit(0);
                }
                if(o.getSource()==times){
```

```
                    disp.setFont(new
Font("TimesRoman",disp.getFont().getStyle(),disp.getFont().getSize()));
                }
                if(o.getSource()==dia){
                    disp.setFont(new
Font("Dialog",disp.getFont().getStyle(),disp.getFont().getSize()));
                }
                if(o.getSource()==sansserif){
                    disp.setFont(new
Font("SansSerif",disp.getFont().getStyle(),disp.getFont().getSize()));
                }
                if(o.getSource()==courier){
                    disp.setFont(new
Font("Courier",disp.getFont().getStyle(),disp.getFont().getSize()));
                }
                if(o.getSource()==bold){
                    disp.setFont(new
Font(disp.getFont().getName(),Font.BOLD,disp.getFont().getSize()));
                }
                if(o.getSource()==italic){
                    disp.setFont(new
Font(disp.getFont().getName(),Font.ITALIC,disp.getFont().getSize()));
                }
                if(o.getSource()==size12){
                    disp.setFont(new
Font(disp.getFont().getName(),disp.getFont().getStyle(),12));
                }
                if(o.getSource()==size14){
                    disp.setFont(new
Font(disp.getFont().getName(),disp.getFont().getStyle(),14));
                }
                if(o.getSource()==size16){
                    disp.setFont(new
Font(disp.getFont().getName(),disp.getFont().getStyle(),16));
```

```
		}
		if(o.getSource()==size18){
			disp.setFont(new
Font(disp.getFont().getName(),disp.getFont().getStyle(),18));
		}
		if(o.getSource()==blu){
			disp.setForeground(new Color(0,0,255));
		}
		if(o.getSource()==bro){
			disp.setForeground(new Color(165,42,42));
		}
		if(o.getSource()==gray){
			disp.setForeground(new Color(128,128,128));
		}
		if(o.getSource()==green){
			disp.setForeground(new Color(0,255,0));
		}
		if(o.getSource()==org){
			disp.setForeground(new Color(255,165,0));
		}
		if(o.getSource()==red){
			disp.setForeground(new Color(255,0,0));
		}
		if(o.getSource()==yel){
			disp.setForeground(new Color(255,255,0));
		}
	if(o.getSource()==india||o.getSource()==aus||o.getSource()==brazil||o.getSource()==germ|
|o.getSource()==ice||o.getSource()==mal||o.getSource()==rus){
		disp.setText(o.getActionCommand());
	}
	}
```

}

2.

Develop Question 1 using JavaFX. Save the file as MenuWithJavaFX.java

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.*;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.control.*;
import javafx.stage.Stage;
import javafx.scene.text.*;
import javafx.scene.text.FontWeight;
import javafx.scene.paint.*;
import javafx.scene.input.*;
public class MenuWithJavaFX extends Application{
        MenuBar mb;
        Menu country,font,name,type,size,colour;
```

```
	MenuItem
times,dia,sansserif,courier,bold,italic,size12,size14,size16,size18,india,aus,brazil,germ,ice,mal,rus,blu
,bro,gray,green,org,red,yel,clc,exit;

	Label disp;

	ContextMenu pop;

	public static void main(String  args[]){

		launch(args);

	}

	public void start(Stage s){

		s.setTitle("creating MenuBar");

		pop= new ContextMenu();

		clc=new MenuItem("Clear");

		clc.setAccelerator(new KeyCodeCombination(KeyCode.C,
KeyCombination.CONTROL_DOWN));

		exit=new MenuItem("Exit");

		exit.setAccelerator(new KeyCodeCombination(KeyCode.E,
KeyCombination.CONTROL_DOWN));

		pop.getItems().addAll(clc,exit);

		mb = new MenuBar();
	country =new Menu("Country");

	country.setAccelerator(new KeyCodeCombination(KeyCode.C, KeyCombination.ALT_DOWN));

	font = new Menu("Font");

	font.setAccelerator(new KeyCodeCombination(KeyCode.F, KeyCombination.ALT_DOWN));

	colour = new Menu("Colour");

	colour.setAccelerator(new KeyCodeCombination(KeyCode.L, KeyCombination.ALT_DOWN));

	name = new Menu("Name");

	name.setAccelerator(new KeyCodeCombination(KeyCode.N, KeyCombination.ALT_DOWN));

	type = new Menu("Type");

	type.setAccelerator(new KeyCodeCombination(KeyCode.T, KeyCombination.ALT_DOWN));

	size = new Menu("Size");

	size.setAccelerator(new KeyCodeCombination(KeyCode.S, KeyCombination.ALT_DOWN));

	times=new MenuItem("TimesRoman");
```

```
        dia=new MenuItem("Dialog");
        sansserif = new MenuItem("SansSerif");
        courier=new MenuItem("Courier");
        bold = new MenuItem("Bold");
        italic = new MenuItem("Italic");
        size12 = new MenuItem("Size 12");
        size14 = new MenuItem("Size 14");
        size16 = new MenuItem("Size 16");
        size18 = new MenuItem("Size 18");
        india = new MenuItem("India");
        aus =new MenuItem("Australia");
        brazil = new MenuItem("Brazil");
        germ = new MenuItem("Germany");
        ice = new MenuItem("Iceland");
        mal = new MenuItem("Malaysia");
        rus = new MenuItem("Russia");
        blu = new MenuItem("Blue");
        bro =new MenuItem("Brown");
        gray = new MenuItem("Gray");
        green = new MenuItem("Green");
        org = new MenuItem("Orange");
        red = new MenuItem("Red");
        yel=new MenuItem("Yellow");
        disp = new Label();
        times.setAccelerator(new KeyCodeCombination(KeyCode.I,
KeyCombination.CONTROL_DOWN));
        dia.setAccelerator(new KeyCodeCombination(KeyCode.D, KeyCombination.CONTROL_DOWN));
        sansserif.setAccelerator(new KeyCodeCombination(KeyCode.A,
KeyCombination.CONTROL_DOWN));
        courier.setAccelerator(new KeyCodeCombination(KeyCode.U,
KeyCombination.CONTROL_DOWN));
```

```
        bold.setAccelerator(new KeyCodeCombination(KeyCode.B,
KeyCombination.CONTROL_DOWN));

        italic.setAccelerator(new KeyCodeCombination(KeyCode.R,
KeyCombination.CONTROL_DOWN));

        size12.setAccelerator(new KeyCodeCombination(KeyCode.F2));

        size14.setAccelerator(new KeyCodeCombination(KeyCode.F4));

        size16.setAccelerator(new KeyCodeCombination(KeyCode.F6));

        size18.setAccelerator(new KeyCodeCombination(KeyCode.F8));

        india.setAccelerator(new KeyCodeCombination(KeyCode.G,
KeyCombination.CONTROL_DOWN));

        aus.setAccelerator(new KeyCodeCombination(KeyCode.H, KeyCombination.CONTROL_DOWN));

        brazil.setAccelerator(new KeyCodeCombination(KeyCode.J,
KeyCombination.CONTROL_DOWN));

        germ.setAccelerator(new KeyCodeCombination(KeyCode.K,
KeyCombination.CONTROL_DOWN));

        ice.setAccelerator(new KeyCodeCombination(KeyCode.M, KeyCombination.CONTROL_DOWN));

        mal.setAccelerator(new KeyCodeCombination(KeyCode.P, KeyCombination.CONTROL_DOWN));

        rus.setAccelerator(new KeyCodeCombination(KeyCode.Q, KeyCombination.CONTROL_DOWN));

        blu.setAccelerator(new KeyCodeCombination(KeyCode.V, KeyCombination.CONTROL_DOWN));

        bro.setAccelerator(new KeyCodeCombination(KeyCode.W, KeyCombination.CONTROL_DOWN));

        gray.setAccelerator(new KeyCodeCombination(KeyCode.X, KeyCombination.CONTROL_DOWN));

        green.setAccelerator(new KeyCodeCombination(KeyCode.Y,
KeyCombination.CONTROL_DOWN));

        org.setAccelerator(new KeyCodeCombination(KeyCode.Z, KeyCombination.CONTROL_DOWN));

        red.setAccelerator(new KeyCodeCombination(KeyCode.F1));

        yel.setAccelerator(new KeyCodeCombination(KeyCode.F3));

        country.getItems().add(india);

        country.getItems().add(aus);

        country.getItems().add(brazil);

        country.getItems().add(germ);

        country.getItems().add(ice);

        country.getItems().add(mal);

        country.getItems().add(rus);
```

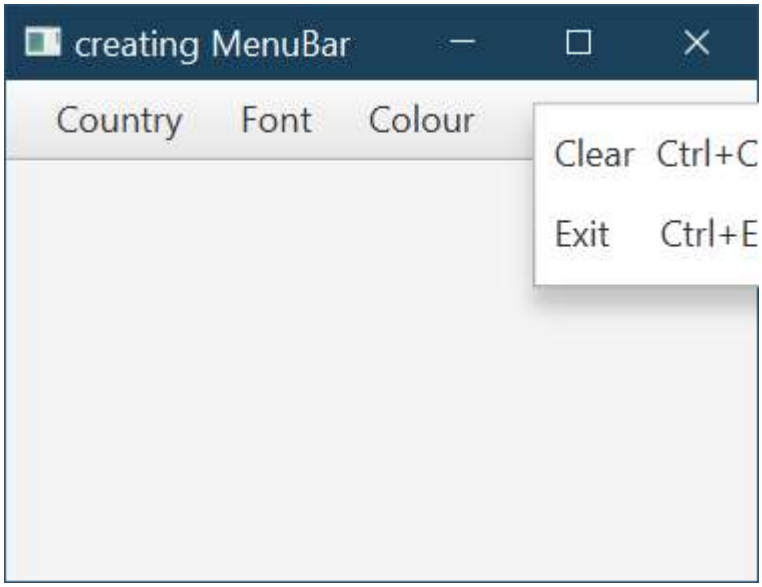
```
mb.getMenus().add(country);
name.getItems().add(times);
name.getItems().add(dia);
name.getItems().add(sansserif);
name.getItems().add(courier);
font.getItems().add(name);
type.getItems().add(bold);
type.getItems().add(italic);
font.getItems().add(type);
size.getItems().add(size12);
size.getItems().add(size14);
size.getItems().add(size16);
size.getItems().add(size18);
font.getItems().add(size);
mb.getMenus().add(font);
colour.getItems().add(blu);
colour.getItems().add(bro);
colour.getItems().add(gray);
colour.getItems().add(green);
colour.getItems().add(org);
colour.getItems().add(red);
colour.getItems().add(yel);
mb.getMenus().add(colour);
EventHandler<ActionEvent> m = new EventHandler<ActionEvent>() {
     Font font;
     FontWeight fw;
     FontPosture fw1;
   public void handle(ActionEvent o)
   {
     if(o.getSource()==clc){
                                    disp.setText("");
```

```
                    }
                    if(o.getSource()==exit){
                            System.exit(0);
                    }
            if(o.getSource()==times){
                FontPosture str = FontPosture.findByName(disp.getFont().getStyle());
                    FontWeight str1=FontWeight.findByName(disp.getFont().getStyle());
                font=Font.font("TimesRoman",str1,str,disp.getFont().getSize());
                            disp.setFont(font);
                    }
                    if(o.getSource()==dia){
                            FontPosture str =
FontPosture.findByName(disp.getFont().getStyle());
                    FontWeight str1=FontWeight.findByName(disp.getFont().getStyle());
                            font=Font.font("Dialog",str1,str,disp.getFont().getSize());
                            disp.setFont(font);
                    }
                    if(o.getSource()==sansserif){
                            FontPosture str =
FontPosture.findByName(disp.getFont().getStyle());
                    FontWeight str1=FontWeight.findByName(disp.getFont().getStyle());
                            font=Font.font("SansSerif",str1,str,disp.getFont().getSize());
                            disp.setFont(font);
                    }
                    if(o.getSource()==courier){
                             FontPosture str =
FontPosture.findByName(disp.getFont().getStyle());
                    FontWeight str1=FontWeight.findByName(disp.getFont().getStyle());
                            font=Font.font("Courier",str1,str,disp.getFont().getSize());
                            disp.setFont(font);
                    }
                    if(o.getSource()==bold){
```

```
                    fw=FontWeight.EXTRA_BOLD;

        disp.setFont(Font.font(disp.getFont().getName(),fw,disp.getFont().getSize()));
                    }
                    if(o.getSource()==italic){
                        fw1=FontPosture.ITALIC;

        disp.setFont(Font.font(disp.getFont().getName(),fw1,disp.getFont().getSize()));
                    }
                    if(o.getSource()==size12){
                        FontPosture str =
FontPosture.findByName(disp.getFont().getStyle());
                       FontWeight str1=FontWeight.findByName(disp.getFont().getStyle());
                        disp.setFont(Font.font(disp.getFont().getName(),str1,str,12));
                    }
                    if(o.getSource()==size14){
                        FontPosture str =
FontPosture.findByName(disp.getFont().getStyle());
                       FontWeight str1=FontWeight.findByName(disp.getFont().getStyle());
                        disp.setFont(Font.font(disp.getFont().getName(),str1,str,14));
                    }
                    if(o.getSource()==size16){
                        FontPosture str =
FontPosture.findByName(disp.getFont().getStyle());
                       FontWeight str1=FontWeight.findByName(disp.getFont().getStyle());
                        disp.setFont(Font.font(disp.getFont().getName(),str1,str,16));
                    }
                    if(o.getSource()==size18){
                        FontPosture str =
FontPosture.findByName(disp.getFont().getStyle());
                       FontWeight str1=FontWeight.findByName(disp.getFont().getStyle());
                        disp.setFont(Font.font(disp.getFont().getName(),str1,str,18));
                    }
```

```
                    if(o.getSource()==blu){
                        disp.setTextFill(Color.web("#0000FF"));
                    }
                    if(o.getSource()==bro){
                        disp.setTextFill(Color.web("#A52A2A"));
                    }
                    if(o.getSource()==gray){
                        disp.setTextFill(Color.web("#808080"));
                    }
                    if(o.getSource()==green){
                        disp.setTextFill(Color.web("#008000"));
                    }
                    if(o.getSource()==org){
                        disp.setTextFill(Color.web("#FFA500"));
                    }
                    if(o.getSource()==red){
                        disp.setTextFill(Color.web("#FF0000"));
                    }
                    if(o.getSource()==yel){
                        disp.setTextFill(Color.web("#FFFF00"));
                    }

        if(o.getSource()==india||o.getSource()==aus||o.getSource()==brazil||o.getSource()==germ||o.getSource()==ice||o.getSource()==mal||o.getSource()==rus){
                        disp.setText(((MenuItem)o.getSource()).getText());
                    }
        }
    };
    india.setOnAction(m);
    aus.setOnAction(m);
    brazil.setOnAction(m);
    germ.setOnAction(m);
```

```
        ice.setOnAction(m);
        mal.setOnAction(m);
        rus.setOnAction(m);
        times.setOnAction(m);
        dia.setOnAction(m);
        sansserif.setOnAction(m);
        courier.setOnAction(m);
        bold.setOnAction(m);
        italic.setOnAction(m);
        size12.setOnAction(m);
        size14.setOnAction(m);
        size16.setOnAction(m);
        size18.setOnAction(m);
        blu.setOnAction(m);
        bro.setOnAction(m);
        gray.setOnAction(m);
        green.setOnAction(m);
        org.setOnAction(m);
        red.setOnAction(m);
        yel.setOnAction(m);
        clc.setOnAction(m);
        exit.setOnAction(m);
        TilePane tilePane = new TilePane(mb);
        mb.setContextMenu(pop);
        VBox vb = new VBox(mb,disp,tilePane);
        Scene sc = new Scene(vb, 300, 200);
        s.setScene(sc);
        s.show();
            }
}
```


creating MenuBar
Country
Font
Colour
Clear Ctrl+C
Exit Ctrl+E