JAVA PROGRAMMING

ASSIGNMENT 5

Name:PUNEETH L

USN: 1BM24MC069

1. Create an abstract class "BankAccount" with abstract methods "deposit()" and "withdraw()". Implement two subclasses "SavingsAccount" and "CheckingAccount" which extend "BankAccount" and implement the abstract methods. Create a "Customer" class which contains a list of "BankAccount" objects. Add methods to the "Customer" class to display account balances, deposit/withdraw money, etc. Create objects of all classes and test their behavior.

PROGRAM:

```
abstract class BankAccount {
    int accno;
    double bal;
    BankAccount(int accno, double bal) {
        this.accno = accno;
        this.bal = bal;
    }
    abstract void deposit(double amount);
    abstract void withdraw(double amount);
    void displayBalance() {
        System.out.println("Account " + accno + " balance: " + bal);
    }
}
class SavingsAccount extends BankAccount {
    SavingsAccount(int accno, double bal) {
        super(accno, bal < 1000 ? 1000 : bal);
        System.out.println("Savings Account created with balance: " + this.bal);
    }
```

```
    @Override
    void deposit(double amount) {
        if (amount < 100) {
            System.out.println("Deposit must be at least 100.");
        } else {
            bal += amount;
            System.out.println("Deposited " + amount + ". New balance: " + bal);
        }
    }
    @Override
    void withdraw(double amount) {
        if (amount > bal - 1000) {
            System.out.println("Withdrawal denied. Minimum balance 1000 must be maintained.");
        } else {
            bal -= amount;
            System.out.println("Withdrew " + amount + ". New balance: " + bal);
        }
    }
}
class CheckingAccount extends BankAccount {
    CheckingAccount(int accno, double bal) {
        super(accno, bal);
        System.out.println("Checking Account created with balance: " + this.bal);
    }
    @Override
    void deposit(double amount) {
        if (amount < 100) {
            System.out.println("Deposit must be at least 100.");
        } else {
```

```
            bal += amount;
            System.out.println("Deposited " + amount + ". New balance: " + bal);
        }
    }
    @Override
    void withdraw(double amount) {
        if (amount > bal) {
            System.out.println("Withdrawal denied. Insufficient balance.");
        } else {
            bal -= amount;
            System.out.println("Withdrew " + amount + ". New balance: " + bal);
        }
    }
}
class Customer {
    String name;
    BankAccount[] accounts;
    int count;
    Customer(String name) {
        this.name = name;
        accounts = new BankAccount[5];
        count = 0;
    }
    void addAccount(BankAccount acc) {
        if (count < accounts.length) {
            accounts[count] = acc;
            count++;
            System.out.println("Account " + acc.accno + " added to customer " + name);
        } else {
            System.out.println("Cannot add more accounts.");
```

```
        }
    }
    void displayBalances() {
        System.out.println("Balances for customer: " + name);
        for (int i = 0; i < count; i++) {
            accounts[i].displayBalance();
        }
    }
    void depositToAccount(int accno, double amount) {
        for (int i = 0; i < count; i++) {
            if (accounts[i].accno == accno) {
                accounts[i].deposit(amount);
                return;
            }
        }
        System.out.println("Account " + accno + " not found.");
    }
    void withdrawFromAccount(int accno, double amount) {
        for (int i = 0; i < count; i++) {
            if (accounts[i].accno == accno) {
                accounts[i].withdraw(amount);
                return;
            }
        }
        System.out.println("Account " + accno + " not found.");
    }
}
```

```
public class Main {


    public static void main(String[] args) {
        Customer cust = new Customer("puneeth");
        SavingsAccount sa = new SavingsAccount(101, 500);
        CheckingAccount ca = new CheckingAccount(102, 2000);
        cust.addAccount(sa);
        cust.addAccount(ca);
        cust.displayBalances();
        cust.depositToAccount(101, 300);
        cust.withdrawFromAccount(102, 500);
        cust.displayBalances();
        cust.withdrawFromAccount(101, 900);
        cust.withdrawFromAccount(101, 200);
        cust.displayBalances();
    }
}
```

OUTPUT:

```
D:\bmsce\2sem\Java programming\5week\aat5>java Main
Savings Account created with balance: 1000.0
Checking Account created with balance: 2000.0
Account 101 added to customer puneeth
Account 102 added to customer puneeth
Balances for customer: puneeth
Account 101 balance: 1000.0
Account 102 balance: 2000.0
Deposited 300.0. New balance: 1300.0
Withdrew 500.0. New balance: 1500.0
Balances for customer: puneeth
Account 101 balance: 1300.0
Account 102 balance: 1500.0
Withdrawal denied. Minimum balance 1000 must be maintained.
Withdrew 200.0. New balance: 1100.0
Balances for customer: puneeth
Account 101 balance: 1100.0
Account 102 balance: 1500.0
```

2. An abstract class called "Marks" is needed to calculate the percentage of marks earned by students A in three subjects (with each subject out of 100) and student B in four subjects (with each subject out of 100). This class must contain the abstract method "getPercentage," which two other classes, "A" and "B," will inherit. The method "getPercentage," which provides the percentage of students, is shared by classes "A" and "B." The constructor of class 'A' will accept the marks obtained in three subjects as its parameters and the constructor of class 'B' will accept the marks obtained in four subjects as its parameters. To test the implementation, objects for both the classes need to be created and the percentage of marks for each student should be printed.

PROGRAM:

```
abstract class Marks {
   int max=100;
   abstract double getPercentage();
}
class A extends Marks {
   int m1, m2, m3;
   A(int m1, int m2, int m3) {
      this.m1 = m1;
      this.m2 = m2;
      this.m3 = m3;
   }
   @Override
   double getPercentage() {
      return (m1 + m2 + m3)*100 / (max*3);
   }
}
class B extends Marks {
   int m1, m2, m3, m4;
   B(int m1, int m2, int m3, int m4) {
      this.m1 = m1;
      this.m2 = m2;
      this.m3 = m3;
```

```
        this.m4 = m4;
    }
    @Override
    double getPercentage() {
        return (m1 + m2 + m3+m4)*100 / (max*4);
    }
}
public class  Student {
    public static void main(String[] args) {
        A s1 = new A(75, 81, 80);
        B s2 = new B(92, 72, 98, 89);


        System.out.println("Percentage of Student A: " + s1.getPercentage() + "%");
        System.out.println("Percentage of Student B: " + s2.getPercentage() + "%");
    }
}
```

OUTPUT:

```
D:\bmsce\2sem\Java programming\5week\aat5>java Student
Percentage of Student A: 78.0%
Percentage of Student B: 87.0%
```

3. Write a Java program using a function root to calculate and display all roots of the quadratic

AX2 + BX + C = 0.

PROGRAM:

```
import java.util.Scanner;
public class QuadraticRoots {
  public static void root(double A, double B, double C) {
     if (A == 0) {
        System.out.println("Coefficient A cannot be zero for a quadratic equation.");
        return;
     }
     double discriminant = B * B - 4 * A * C;
     if (discriminant > 0) {
        double root1 = (-B + Math.sqrt(discriminant)) / (2 * A);
        double root2 = (-B - Math.sqrt(discriminant)) / (2 * A);
        System.out.println("Roots are real and distinct:");
        System.out.println("Root 1 = " + root1);
        System.out.println("Root 2 = " + root2);
     } else if (discriminant == 0) {
        double root = -B / (2 * A);
        System.out.println("Roots are real and equal:");
        System.out.println("Root = " + root);
     } else {
        double realPart = -B / (2 * A);
        double imaginaryPart = Math.sqrt(-discriminant) / (2 * A);
        System.out.println("Roots are complex and imaginary:");
        System.out.println("Root 1 = " + realPart + " + " + imaginaryPart + "i");
        System.out.println("Root 2 = " + realPart + " - " + imaginaryPart + "i");
     }
  }
```

```
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter coefficient A:");
        double A = scanner.nextDouble();
        System.out.println("Enter coefficient B:");
        double B = scanner.nextDouble();
        System.out.println("Enter coefficient C:");
        double C = scanner.nextDouble();
        root(A, B, C);
    }
}
```

OUTPUT:

```
D:\bmsce\2sem\Java programming\5week\aat5>java QuadraticRoots
Enter coefficient A:
2
Enter coefficient B:
4
Enter coefficient C:
2
Roots are real and equal:
Root = -1.0
```

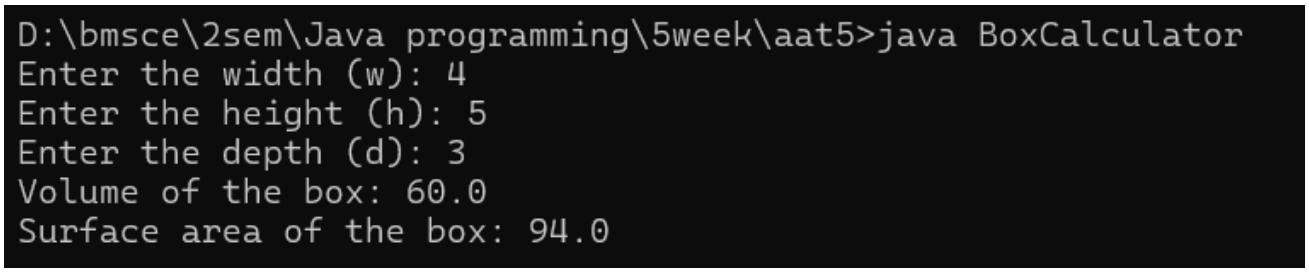
4. Write a program to find the volume of a box that has its sides w, h, d as width, height, and depth, respectively. Its volume is v = w * h * d and also fi nd the surface area given by the formula s = 2(wh + hd + dw).

PROGRAM:

```
import java.util.Scanner;
class Box{
   double w,h,d;
   Box(double w,double h,double d){
      this.w=w;
      this.h=h;
      this.d=d;
   }
   public  double calculateVolume() {
      return w * h * d;
   }
   public double calculateSurfaceArea() {
      return 2 * (w * h + h * d + d * w);
   }
}
public class BoxCalculator {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter the width (w): ");
      double w = scanner.nextDouble();
      System.out.print("Enter the height (h): ");
      double h = scanner.nextDouble();
      System.out.print("Enter the depth (d): ");
      double d = scanner.nextDouble();
      Box b1= new Box(w,h,d);
      double volume = b1.calculateVolume();
      double surfaceArea = b1.calculateSurfaceArea();
```

```
        System.out.println("Volume of the box: " + volume);
        System.out.println("Surface area of the box: " + surfaceArea);
        scanner.close();
    }
}
```

OUTPUT:



5. A class weight is having a data member pound, which will have the weight in pounds. Using a conversion function, convert the weight in pounds to weight inkilograms which is of double type. Write a program to do this. 1 pounds =1 kg/0.453592

Use default constructor to initial assignment of 1000 pounds.

PROGRAM:

```
public class Weight {
   private int pound;
   public Weight() {
      this.pound = 1000;}
   public Weight(int pound) {
      this.pound = pound;}
   public double toKilograms() {
      return pound * 0.453592;}
   public int getPound() {
      return pound;}
   public void setPound(int pound) {
      this.pound = pound;}
   public static void main(String[] args) {
      Weight w = new Weight();
      System.out.println("Weight in pounds: " + w.getPound());
      System.out.println("Weight in kilograms: " + w.toKilograms());
      w.setPound(500);
      System.out.println("New Weight in pounds: " + w.getPound());
      System.out.println("New Weight in kilograms: " + w.toKilograms());
   }}
```

OUTPUT:

```
D:\bmsce\2sem\Java programming\5week\aat5>javac Weight.java

D:\bmsce\2sem\Java programming\5week\aat5>java Weight
Weight in pounds: 1000
Weight in kilograms: 453.592
New Weight in pounds: 500
New Weight in kilograms: 226.796
```