Table of Contents

Data types and operators

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

            float interest = (3250*7*3)/100.0F;
            System.out.println(interest);
      }
}
```

Selection Control Structure

```
class Tester {
      public static void main(String[] args) {
            // Implement your code here
            int num1=3, num2=4, num3=1;

            if(num1>num2) {
                if(num3>num1) {
                    System.out.println(num3);
                }
                else
                System.out.println(num1);
            }
            else {
                if(num2>num3) {
                    System.out.println(num2);
                }
                else
                System.out.println(num3);
            }
      }
}
```

Introduction to Java

```
class Tester {
      public static void main(String[] args) {
            System.out.println("Door No: D089");
            System.out.println("Street: St. Louis Street");
            System.out.println("City: Springfield");
            System.out.println("ZIP Code: 62729");
      }
}
```

Data types and Operators

```
class Tester {
	public static void main(String[] args) {
		int radius = 4;
		System.out.println(3.14F*radius*radius);
	}
}
```

Data types and Operators

```
class Tester {
	public static void main(String[] args) {
		int F = 32;
		
		System.out.println((float)((F-32)/9)*5);
	}
}
```

Selection Control Structure 1

```
class Tester {
	public static void main(String[] args) {
		int num1=5, num2=5;
		
		if(num1==num2)
		System.out.println(num1+num2);
		else
		System.out.println(2*(num1+num2));
	}
}
```

Selection Control Structure 2

```
class Tester {
	public static void main(String[] args) {
		int a=1, b=4, c=4;
		
		float dis = (b*b) - (4*a*c);
		
		if(dis==0) {
```

```
                float x = (-b) / (2*a);
                System.out.println("The root is "+x);
            }
            else if(dis>0) {
                float x1 = (-b+dis) / (2*a);
                float x2 = (-b-dis) / (2*a);
                System.out.println("The roots are "+x1+","+x2);
            }
            else {
                System.out.println("No real roots");
            }
      }
}
```

Selection Control Structure 3

```
class Tester {
      public static void main(String[] args) {
            int num1=1, num2=5, num3=3;

            int p;
            if(num1==7) {
                p = num2*num3;
            }
            else if(num2==7) {
                p = num3;
            }
            else if(num3==7) {
                p = -1;
            }
            else {
                p = num1*num2*num3;
            }

            System.out.println(p);
      }
}
```

Selection Control Structure 1

```
class Tester {
      public static void main(String[] args) {
            char typeOfFood = 'V';
            int distance = 7;
            int quantity = 1;

            int totalCost = 0;

            if(quantity<1) {//exit program
            }

            if(typeOfFood=='V') {
                totalCost += 12*quantity;
            }
            else {
                totalCost += 15*quantity;
            }

            if(distance<=3) {
                totalCost += 0;
            }
            else if(distance<=6) {
                totalCost += (distance-3)*1;
            }
            else {
                totalCost += 3 + (distance-6)*2;
            }

            System.out.println(totalCost);
      }
}
```

Selection Control Structure 2

```
class Tester {
      public static void main(String[] args) {
            int accNo=1001, salary=40000, accBal=250000;
            String loanType="Car";
            int loanAmtExpected=300000, emisExpected=30;

            // acc num validity
            if(accNo-1000>=0) { //valid num of digiits
                if(accNo%1000 == 1) { // valid
                }
                else {
                    // do whatever
                }
            }
            else {
                // do whatever
            }
            
            if(salary>25000) {
                if(loanType=="Car") {
                    if(loanAmtExpected<=500000 && emisExpected<=36) {
                        System.out.println("Loan granted");
                    }
                }
            }
            else if(salary>50000) {
                if(loanType=="House") {
                    if(loanAmtExpected<=6000000 && emisExpected<=60) {
                        System.out.println("Loan granted");
                    }
                }
            }
            else if(salary>75000) {
                if(loanType=="Business") {
                    if(loanAmtExpected<=7500000 && emisExpected<=84) {
                        System.out.println("Loan granted");
                    }
                }
            }
            else {
                System.out.println("Loan failed");
            }
      }
}
```

Iteration Control Structure 1

```
class Tester {
      public static void main(String[] args) {
            int num=5;
            int p=1;
            for(int i=num; i>=1; i--) {
                p*=i;
            }
            System.out.println(p);
      }
}
```

Iteration Control structure 2

```
class Tester {
      public static void main(String[] args) {
            int n=5;
            int p=1;
            for(int i=1; i<=n; i++) {
                System.out.println(p);
                p *= 2;
            }
      }
}
```

Methods 1

```
class Calculator {

      public double findAverage(int number1, int number2, int number3) {
          double x = ((number1+number2+number3)/3.0);
          return(Math.round(x * 100.0) / 100.0);
      }
}

class Tester {

      public static void main(String args[]) {
            Calculator calculator = new Calculator();
            // Invoke the method findAverage of the Calculator class and
display the average
            System.out.println(calculator.findAverage(12,8,15));
      }
}
```

Iteration control structure 1

```
class Tester {
      public static void main(String[] args) {
            int r,sum=0,temp;
```

```
        int n=1331;

        temp=n;
        while(n>0){
         r=n%10;  //getting remainder
         sum=(sum*10)+r;
         n=n/10;
          }
        if(temp==sum)
         System.out.println("palindrome number ");
        else
         System.out.println("not palindrome");
            }
}
```

Iteration control structure 2

```
class Tester {
      public static void main(String[] args) {
            int heads=150, legs=500;

            if(legs%2!=0) { //invalid
            }

            int chickens = (legs - 2*heads) / 4;
            int rabbits = heads - chickens;

            System.out.println("Chickens = "+chickens+",Rabbits = 
"+rabbits);
      }
}
```

Iteration control structure 3

```
class Tester {
      public static void main(String[] args) {
            int num1=2250;
            int num=num1, sum=0;

            while(num>0) {
                int l = num%10;
                num = num/10;
                sum += l;
            }

            if(num1%sum == 0) {System.out.println("Divisible");}
            else {System.out.println("Not Divisible");}
      }
}
```

Iteration control structure 4

```
class Tester {
      public static void main(String[] args) {
            int number = 371, originalNumber, remainder, result = 0;

        originalNumber = number;

        while (originalNumber != 0)
        {
            remainder = originalNumber % 10;
            result += Math.pow(remainder, 3);
            originalNumber /= 10;
        }

        if(result == number)
            System.out.println(number + " is an Armstrong number.");
        else
            System.out.println(number + " is not an Armstrong number.");
      }
}
```

Encapsulation - 1

```
class Employee {

      private String employeeId;
      private String employeeName;
      private int salary;
      private int bonus;
      private int jobLevel;

      public void calculateSalary() {
            if (this.jobLevel >= 4) {
                  this.bonus = 100;
            } else {
                  this.bonus = 50;
            }
            this.salary += this.bonus;
      }

      public void setEmployeeId(String employeeId) {
          this.employeeId = employeeId;
      }
      public void setEmployeeName(String employeeName) {
          this.employeeName = employeeName;
      }
      public void setSalary(int salary) {
          this.salary = salary;
      }
      public void setBonus(int bonus) {
          this.bonus = bonus;
      }
      public void setJobLevel(int jobLevel) {
          this.jobLevel = jobLevel;
      }

      public String getEmployeeId() {
          return this.employeeId;
      }
      public String getEmployeeName() {
          return this.employeeName;
      }
      public int getSalary() {
          return this.salary;
      }
      public int getBonus() {
          return this.bonus;
      }
      public int getJobLevel() {
          return this.jobLevel;
      }
}

class Tester {

      public static void main(String args[]) {

            Employee employee = new Employee();
            //employee.employeeId = "C101";
            employee.setEmployeeId("C101");
            //employee.employeeName = "Steve";
            employee.setEmployeeName("Steve");
            //employee.salary = 650;
            employee.setSalary(650);
            //employee.jobLevel = 4;
            employee.setJobLevel(4);

            employee.calculateSalary();

            System.out.println("Employee Details");
            System.out.println("Employee Id: " +
employee.getEmployeeId());
            System.out.println("Employee Name: " +
employee.getEmployeeName());
            System.out.println("Salary: " + employee.getSalary());

      }
}
```

Array - 1

```
class Tester {

    public static int calculateSumOfEvenNumbers(int[] numbers){
        //Implement your code here and change the return value
accordingly
        int len = numbers.length;
        int sum=0;
        for(int i=0; i<len; i++) {
            if(numbers[i]%2 == 0) {
                sum += numbers[i];
            }
        }

        return sum;
    }

      public static void main(String[] args) {
            int[] numbers = {68,79,86,99,23,2,41,100};
            System.out.println("Sum of even numbers: "
+calculateSumOfEvenNumbers(numbers));
      }
}
```

String - 1

```
class Tester{

    public static String removeWhiteSpaces(String str){
            //Implement your code here and change the return value
accordingly
            int len = str.length();
            String str1 = "";
            for(int i=0; i<len; i++) {
                if(str.charAt(i)!=' ') {
                    str1+= str.charAt(i);
                }
            }
        return str1;
      }

      public static void main(String args[]){
            String str = "Hello   How are you   ";
            str = removeWhiteSpaces(str);
            System.out.println(str);
      }
}
```

Array - 1

```
class Teacher {
    //Implement your code here
    String teacherName;
    String subject;
    double salary;

    public Teacher(String teacherName, String subject, double salary) {
        this.teacherName = teacherName;
        this.subject = subject;
        this.salary = salary;
    }

    public String getTeacherName() {
        return this.teacherName;
    }
    public void setTeacherName(String teacherName) {
        this.teacherName = teacherName;
    }

    public String getSubject() {
        return this.subject;
    }
    public void setSubject(String subject) {
        this.subject = subject;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }
    public double getSalary() {
        return this.salary;
    }

    public void display() {
        System.out.println("Name : "+teacherName+", Subject :
"+subject+", Salary : "+salary);
    }
}

class Tester {
      public static void main(String[] args) {
          // Implement your code here
          Teacher[] teachers = new Teacher[4];
          teachers[0] = new Teacher("Alex", "Java Fundamentals", 1200);
          teachers[1] = new Teacher("Jon", "RDBMS", 800);
          teachers[2] = new Teacher("Sam", "Networking", 900);
          teachers[3] = new Teacher("Maria", "Python", 900);

          for(int i=0; i<4; i++) {
              teachers[i].display();
          }
      }
}
```

Array - 2

```
class Tester {

    public static double[] findDetails(double[] salary) {
        //Implement your code here and change the return value
accordingly
        double[] ret = new double[3];
        double sum=0;
        for(int i=0; i<salary.length; i++) {
            sum += salary[i];
        }
        double avg = sum / salary.length;
        double n1=0, n2=0;
        for(int i=0; i<salary.length; i++) {
            if(salary[i] > avg) {n1++;}
            else if(salary[i]<avg) {n2++;}
        }
        ret[0] = avg;
        ret[1] = n1;
        ret[2] = n2;
        return ret;

    }

    public static void main(String[] args) {
        double[] salary = { 23500.0, 25080.0, 28760.0, 22340.0, 19890.0
};
        double[] details = findDetails(salary);

        System.out.println("Average salary: "+ details[0]);
        System.out.println("Number of salaries greater than the average
salary: "+ details[1]);
        System.out.println("Number of salaries lesser than the average
salary: "+ details[2]);
    }
}
```

Array 3

```
class Tester {

   public static int[] findLeapYears(int year){
       //Implement your code here and change the return value
accordingly
       int[] ret = new int[15];
       for(int i=0; i<15; i++) {
           while(!(isLeapYear(year))) {year++;}
           ret[i] = year;
           year++;
       }
        return ret;
   }
   public static boolean isLeapYear(int year) {
       return (((year % 4 == 0) && (year % 100!= 0)) || (year%400 ==
0));
   }

   public static void main(String[] args) {
       int year = 2000;
         int[] leapYears;
         leapYears=findLeapYears(year);
         for ( int index = 0; index<leapYears.length; index++ ) {
             System.out.println(leapYears[index]);
         }
    }
}
```

Array 4

```
class Tester {

      public static int[] findNumbers(int num1, int num2) {
            int[] numbers = new int[6];
            int k=0;

            // Implement your code here
            if(num1 < num2) {
                for(int i=num1+1; i<num2; i++) {
                    // if two digit numbers
                    if(i>=10 && i<=99) {
                        if(cond(i)) {
                            numbers[k++] = i;
                        }
                    }
                }
            }

            return numbers;
      }

      public static boolean cond(int num) {
          boolean c1, c2;

          if((num%10 + num/10)%3 == 0) c1 = true;
          else c1 = false;

          if(num%5 == 0) c2 = true;
          else c2 = false;

          return(c1 && c2);
      }

      public static void main(String[] args) {
            int num1 = 10;
            int num2 = 30;

            int[] numbers = findNumbers(num1, num2);
            if (numbers[0] == 0) {
                  System.out.println("There is no such number!");
            } else {
                  for (int index = 0; index <= numbers.length - 1;
index++) {
                        if (numbers[index] == 0) {
                              break;
                        }
                        System.out.println(numbers[index]);
                  }
            }

      }
}
```

String 1

```
class Tester{

    public static String moveSpecialCharacters(String str){
            //Implement your code here and change the return value
accordingly
            String str1="", str2="";
            for(int i=0; i<str.length(); i++) {
                char ch = str.charAt(i);
                if((ch >= 97 && ch <= 122) || (ch >= 65 && ch <= 90)) {
                    str1 += ch;
                }
                else
                str2 += ch;
            }
        return str1.concat(str2);
      }

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

```
     String str = "He@#$llo!*&";
     System.out.println(moveSpecialCharacters(str));
   }

}
```

String 2

```
class Tester{
  public static boolean checkPalindrome(String str){
     int i = 0, j = str.length() - 1;

    while (i < j) {

      if (str.charAt(i) != str.charAt(j))
        return false;

      i++;
      j--;
    }

    return true;
   }

   public static void main(String args[]){
      String str = "radar";
      if(checkPalindrome(str))
         System.out.println("The string is a palindrome!");
      else
         System.out.println("The string is not a palindrome!");
   }
}
```

String 3

```
class Tester {
  public static String reverseEachWord(String str){
     String[] words = str.split(" ");
   String reversedString = "";
   for (int i = 0; i < words.length; i++)
    {
     String word = words[i];
     String reverseWord = "";
```

```
           for (int j = word.length()-1; j >= 0; j--)
         {
            reverseWord = reverseWord + word.charAt(j);
         }
         reversedString = reversedString + reverseWord + " ";
      }
      String str1 = reversedString;
      str1 = str1.substring(0, str1.length() - 1);
      return(str1);
      }

      public static void main(String args[]){
          String str = "all cows eat grass";
          System.out.println(reverseEachWord(str));
      }
}
```

Static 1

```
class Bill{
    private static int counter;
    private String billId;
    private String paymentMode;

    static {
        counter=9001;
    }

    public Bill(String paymentMode){
        this.paymentMode = paymentMode;
    }

    public void setBillId(String billId) {
        this.billId = billId;
    }
    public String getBillId() {
        int c = counter;
        counter++;
        return("B"+c);
    }
    public String getPaymentMode() {
        return this.paymentMode;
    }
    public void setPaymentMode(String paymentMode) {
        this.paymentMode = paymentMode;
    }
    public static int getCounter() {
        return(counter);
    }
}

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

        Bill bill1 = new Bill("DebitCard");
        Bill bill2 = new Bill("PayPal");

        //Create more objects and add them to the bills array for
testing your code

        Bill[] bills = { bill1, bill2 };

        for (Bill bill : bills) {
            System.out.println("Bill Details");
            System.out.println("Bill Id: " + bill.getBillId());
            System.out.println("Payment method: " +
bill.getPaymentMode());
            System.out.println();
       }
    }
}
```

Association - 1

```
class CabServiceProvider{
    private String cabServiceName;
    private int totalCabs;

    public CabServiceProvider(String cabServiceName, int totalCabs) {
        this.cabServiceName = cabServiceName;
        this.totalCabs = totalCabs;
    }

    public void setCabServiceName(String cabServiceName) {
        this.cabServiceName = cabServiceName;
    }
    public String getCabServiceName() {
        return this.cabServiceName;
    }
    public void setTotalCabs(int totalCabs) {
        this.totalCabs = totalCabs;
    }
    public int getTotalCabs() {
        return this.totalCabs;
    }

    public double calculateRewardPrice(Driver driver) {
        double bonus;
        if(this.cabServiceName.equals("Halo")) {
            float r = driver.getAverageRating();

            if(r>=4.5 && r<=5) {
                bonus = 10*r;
            }
            else if(r>=4 && r<4.5) {
                bonus = 5*r;
            }
            else
            bonus=0F;
        }
        else if(this.cabServiceName.equals("Aber")) {
            float r = driver.getAverageRating();

            if(r>=4.5 && r<=5) {
                bonus = 8*r;
            }
            else if(r>=4 && r<4.5) {
                bonus = 3*r;
            }
            else bonus=0F;
        }
        else
        bonus=0F;

        return(Math.round(bonus * 100.0) / 100.0);
    }
}

class Driver {

      private String driverName;
      private float averageRating;

      public Driver(String driverName, float averageRating){
            this.driverName=driverName;
            this.averageRating=averageRating;
      }

      public String getDriverName(){
            return this.driverName;
      }

      public void setDriverName(String driverName){
            this.driverName=driverName;
      }

      public float getAverageRating(){
            return this.averageRating;
      }

      public void setAverageRating(float averageRating){
            this.averageRating=averageRating;
      }

    //DO NOT MODIFY THE METHOD
    //Your exercise might not be verified if the below method is
modified
    public String toString(){
        return "Driver\ndriverName: "+this.driverName+"\naverageRating:
"+this.averageRating;
    }
}

class Tester {

      public static void main(String args[]){
          CabServiceProvider cabServiceProvider1 = new
CabServiceProvider("Halo", 50);

            Driver driver1 = new Driver("Luke", 4.8f);
            Driver driver2 = new Driver("Mark", 4.2f);
            Driver driver3 = new Driver("David", 3.9f);

            Driver[] driversList = { driver1, driver2, driver3 };
            for (Driver driver : driversList) {
                  System.out.println("Driver Name:
"+driver.getDriverName());
                  double bonus =
cabServiceProvider1.calculateRewardPrice(driver);
                  if (bonus>0)
                        System.out.println("Bonus: $"+bonus+"\n");
                  else
                        System.out.println("Sorry, bonus is not 
available!");
            }

            //Create more objects of CabServiceProvider and Driver 
classes for testing your code
      }
}
```

Static 2

```
class Participant {
    private static int counter;
    static {
        counter=10001;
    }
    private String registrationId;
    private String name;
    private long contactNumber;
    private String city;

    public Participant(String name, long contactNumber, String city) {
        this.name = name;
        this.contactNumber = contactNumber;
        this.city = city;
    }
    public String getRegistrationId() {
        this.registrationId = "D" + counter;
        counter++;
        return this.registrationId;
    }
    public static int getCounter() {
        return counter;
    }
    public static void setCounter(int counter) {
        counter = counter;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getName() {
        return this.name;
    }
    public void setCity(String city) {
        this.city = city;
    }
    public String getCity() {
        return this.city;
    }
    public void setContactNumber(long contactNumber) {
        this.contactNumber = contactNumber;
    }
    public long getContactNumber() {
        return this.contactNumber;
    }

}

class Tester {

      public static void main(String[] args) {

            Participant participant1 = new Participant("Franklin",
7656784323L, "Texas");
            Participant participant2 = new Participant("Merina",
7890423112L, "New York");

            //Create more objects and add them to the participants array
for testing your code

            Participant[] participants = { participant1, participant2 };

            for (Participant participant : participants) {
                  System.out.println("Hi "+participant.getName()+"! Your
registration id is "+participant.getRegistrationId());
            }

      }
}
```

Static 3

```
class Booking{
    //Implement your code here
    private String customerEmail;
    private int seatsRequired;
    private boolean isBooked;
    private static int seatsAvailable;
    static {
        seatsAvailable = 400;
    }

    public Booking(String customerEmail, int seatsRequired) {
        this.customerEmail = customerEmail;
        this.seatsRequired = seatsRequired;
        if(seatsRequired <= seatsAvailable) {
            seatsAvailable -= seatsRequired;
            isBooked = true;
        }
        else {
            isBooked = false;
        }
    }
    public String getCustomerEmail() {
        return this.customerEmail;
    }
    public void setCustomerEmail(String customerEmail) {
        this.customerEmail = customerEmail;
    }
    public int getSeatsRequired() {
        return this.seatsRequired;
    }
    public void setSeatsRequired(int seatsRequired) {
        this.seatsRequired = seatsRequired;
    }
    public static int getSeatsAvailable() {
        return seatsAvailable;
    }
    public static void setSeatsAvailable(int seatsAvailable) {
        seatsAvailable = seatsAvailable;
    }
    public boolean isBooked() {
       return this.isBooked;
    }
    public void setBooked(Boolean isBooked) {
        this.isBooked = isBooked;
    }
}

class Tester {
    public static void main(String[] args) {
        Booking booking1 = new Booking("jack@email.com", 100);
        Booking booking2 = new Booking("jill@email.com", 350);
        //Create more objects and add them to the bookings array for
testing your code

        Booking[] bookings = { booking1, booking2 };

        for (Booking booking : bookings) {
            if (booking.isBooked()) {
                System.out.println(booking.getSeatsRequired()+" seats
successfully booked for "+booking.getCustomerEmail());
            }
            else {
                System.out.println("Sorry
"+booking.getCustomerEmail()+", required number of seats are not
available!");
                System.out.println("Seats available:
"+Booking.getSeatsAvailable());
            }
         }
    }
}
```

Inheritance - 1

```
class Camera {
      private String brand;
      private double cost;

      public Camera() {
            this.brand = "Nikon";
      }

      public String getBrand() {
            return brand;
      }
      public void setBrand(String brand) {
            this.brand = brand;
      }
      public double getCost() {
            return cost;
      }
      public void setCost(double cost) {
            this.cost = cost;
      }
}

class DigitalCamera extends Camera {
      private int memory;

      public DigitalCamera(String brand, double cost) {
            this.memory = 16;
            super.setBrand(brand);
            super.setCost(cost);
      }

    public int getMemory() {
            return memory;
      }
      public void setMemory(int memory) {
            this.memory = memory;
      }
}

class Tester {
      public static void main(String[] args) {
        DigitalCamera camera = new DigitalCamera("Canon",100);
        System.out.println(camera.getBrand()+" "+camera.getCost()+"
"+camera.getMemory());
    }
}
```

Method overloading- 1

```
class Point {
    //Implement your code here
    private double xCoordinate;
    private double yCoordinate;

    public Point(double xCoordinate, double yCoordinate) {
        this.xCoordinate = xCoordinate;
        this.yCoordinate = yCoordinate;
    }
    public double calculateDistance() {
        double result = Math.sqrt(xCoordinate*xCoordinate +
yCoordinate*yCoordinate);
        return(Math.round(result * 100.0) / 100.0);
    }
    public double calculateDistance(Point point) {
        double result = Math.sqrt(Math.pow(this.xCoordinate-
point.xCoordinate, 2.0) + Math.pow(this.yCoordinate-point.yCoordinate,
2.0));
        return(Math.round(result * 100.0) / 100.0);
    }
    public double getxCoordinate() {
        return xCoordinate;
    }

    public void setxCoordinate(double xCoordinate) {
        this.xCoordinate = xCoordinate;
    }

    public double getyCoordinate() {
        return yCoordinate;
    }

    public void setyCoordinate(double yCoordinate) {
        this.yCoordinate = yCoordinate;
    }
}


class Tester {

      public static void main(String[] args) {
            Point point1 = new Point(3.5, 1.5);
        Point point2 = new Point(6, 4);

        System.out.println("Distance of point1 from origin is
"+point1.calculateDistance());
        System.out.println("Distance of point2 from origin is
"+point2.calculateDistance());
        System.out.println("Distance of point1 from point2 is
"+point1.calculateDistance(point2));

        //Create more objects for testing your code

      }
}
```

Aggregation - 2

```
class Author {
    private String name;
    private String emailid;
    private char gender;

    public Author(String name, String emailid, char gender) {
        this.name = name;
        this.emailid = emailid;
        this.gender = gender;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmailid() {
        return emailid;
    }

    public void setEmailid(String emailid) {
        this.emailid = emailid;
    }

    public char getGender() {
        return gender;
    }

    public void setGender(char gender) {
        this.gender = gender;
    }
}

class Book {
    private String name;
    private Author author;
    private Double price;
    private int quantity;

    public Book(String name, Author author, Double price, int quantity)
{
        this.name = name;
        this.author = author;
        this.price = price;
        this.quantity = quantity;
    }
```

```java
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Author getAuthor() {
        return author;
    }

    public void setAuthor(Author author) {
        this.author = author;
    }

    public Double getPrice() {
        return price;
    }

    public void setPrice(Double price) {
        this.price = price;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
    public void displayAuthorDetails() {
        System.out.println("Author name: " + this.author.getName());
        System.out.println("Author email: " + this.author.getEmailid());
        System.out.println("Author gender: " + this.author.getGender());
    }
}

class Tester {
    public static void main(String[] args) {
        //Implement your code here
    }
}
```

Aggregation - 3

```
class Room {
    //Implement your code here
    private int roomNo;
    private int capacity;
    private static int roomCounter;
    private int cap;

    static {
        roomCounter = 500;
    }

    public Room() {
        this.capacity = 4;
        this.roomNo = roomCounter + 1;
        cap=0;
    }

    public int getCap() {
        return this.cap;
    }

    public int getCapacity() {
        return this.capacity;
    }

    public void setCapacity(int capacity) {
        this.capacity = capacity;
    }

    public static int getRoomCounter() {
        return roomCounter;
    }

    public static void setRoomCounter(int roomCounter) {
        Room.roomCounter = roomCounter;
    }

    public int getRoomNo() {
        return roomNo;
    }

    //Uncomment the below method after implementation before verifying
    //DO NOT MODIFY THE METHOD
    /*public String toString(){
        return "Room\nroomNo: "+this.roomNo+"\ncapacity:
"+this.capacity;
    }*/

}

class Member {
      //Implement your code here
      private int memberId;
    private String name;
    private Room room;

    public Member(int memberId, String name) {
        this.memberId = memberId;
        this.name = name;
    }

    public int getMemberId() {
        return memberId;
    }

    public void setMemberId(int memberId) {
        this.memberId = memberId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Room getRoom() {
        return room;
    }

    public void setRoom(Room room) {
        this.room = room;
    }

      //Uncomment the below method after implementation before verifying
    //DO NOT MODIFY THE METHOD

    /*public String toString(){
        return "Member\nmemberId: "+this.memberId+"\nname: "+this.name;
    }*/
}

class Admin {
    public void assignRoom(Room[] rooms, Member member) {
        //System.out.println("For: " + member.getName());
        for(int i=0; i<rooms.length; i++) {
            System.out.println("Checking room"+i+",
"+rooms[i].getCap());
            if(rooms[i].getCap() < 4) {
                System.out.println("Valid room: "+rooms[i].getRoomNo());
                // this room can be allocated
                rooms[i].setCapacity(rooms[i].getCapacity() + 1);
                member.setRoom(rooms[i]);
            }
            else {
                Room.setRoomCounter(Room.getRoomCounter()+1);
            }
        }
    }
}


class Tester {
      public static void main(String args[]) {
            Room room1 = new Room();
            Room room2 = new Room();
            Room room3 = new Room();
            Room room4 = new Room();
            Room room5 = new Room();

            Room[] totalRooms = { room1, room2, room3, room4, room5 };

            Admin admin = new Admin();

            Member member1 = new Member(101, "Serena");
            Member member2 = new Member(102, "Martha");
            Member member3 = new Member(103, "Nia");
            Member member4 = new Member(104, "Maria");
            Member member5 = new Member(105, "Eva");

            Member[] members = { member1, member2, member3, member4,
member5 };

            for (Member member : members) {
                  admin.assignRoom(totalRooms, member);
                  if(member.getRoom()!=null) {
                        System.out.println("Hi "+member.getName()+"! Your 
room number is "+member.getRoom().getRoomNo());
                  }
                  else {
                        System.out.println("Hi "+member.getName()+"! No 
room available");
                  }
            }
      }
}
```

Inheritance - 1

```
class Employee {

    private int employeeId;
    private String employeeName;
    private double salary;

    public Employee(int employeeId, String employeeName) {
        this.employeeId = employeeId;
        this.employeeName = employeeName;
    }

    public int getEmployeeId() {
        return employeeId;
    }

    public void setEmployeeId(int employeeId) {
        this.employeeId = employeeId;
    }

    public String getEmployeeName() {
        return employeeName;
    }

    public void setEmployeeName(String employeeName) {
        this.employeeName = employeeName;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }
    //Uncomment the below method after implementation before verifying
    //DO NOT MODIFY THE METHOD

    public String toString(){
        return "Employee\nemployeeId: 
"+this.getEmployeeId()+"\nemployeeName: 
"+this.getEmployeeName()+"\nsalary: "+this.getSalary();
    }
}


class PermanentEmployee extends Employee {

    private double basicPay;
    private double hra;
    private float experience;

    public PermanentEmployee(int empId, String name, double basicPay, 
double hra, float experience) {
        super(empId, name);
        this.basicPay = basicPay;
        this.hra = hra;
        this.experience = experience;
    }

    public double getBasicPay() {
        return basicPay;
    }

    public void setBasicPay(double basicPay) {
        this.basicPay = basicPay;
    }

    public double getHra() {
        return hra;
    }

    public void setHra(double hra) {
        this.hra = hra;
    }
    public float getExperience() {
        return experience;
    }

    public void setExperience(float experience) {
        this.experience = experience;
    }

    public void calculateMonthlySalary() {
        if(this.experience < 3) {
            super.setSalary(this.basicPay + this.hra);
        }
        else if(this.experience >= 3 && this.experience <5) {
            super.setSalary(this.basicPay * 1.05  + this.hra);
        }
        else if(this.experience >= 5 && this.experience <10) {
            super.setSalary(this.basicPay * 1.07  + this.hra);
        }
        else if(this.experience >= 10) {
            super.setSalary(this.basicPay * 1.1  + this.hra);
        }
    }

    //Uncomment the below method after implementation before verifying
    //DO NOT MODIFY THE METHOD

    public String toString(){
        return "PermanentEmployee\nemployeeId: 
"+this.getEmployeeId()+"\nemployeeName: 
"+this.getEmployeeName()+"\nsalary: "+this.getSalary()+"\nbasicPay: 
"+this.getBasicPay()+"\nhra: "+this.getHra()+"\nexperience: 
"+this.getExperience();
    }
}

class ContractEmployee extends Employee {

    private double wage;
    private float hoursWorked;

    public ContractEmployee(int empId, String name, double wage, float 
hoursWorked) {
        super(empId, name);
        this.wage = wage;
        this.hoursWorked = hoursWorked;
    }
    public double getWage() {
        return wage;
    }

    public void setWage(double wage) {
        this.wage = wage;
    }

    public float getHoursWorked() {
        return hoursWorked;
    }

    public void setHoursWorked(float hoursWorked) {
        this.hoursWorked = hoursWorked;
    }
    public void calculateSalary() {
        super.setSalary(hoursWorked * wage);
    }

    //Uncomment the below method after implementation before verifying
    //DO NOT MODIFY THE METHOD

    public String toString(){
        return "ContractEmployee\nemployeeId: 
"+this.getEmployeeId()+"\nemployeeName: 
"+this.getEmployeeName()+"\nsalary: "+this.getSalary()+"\nwage: 
"+this.getWage()+"\nhoursWorked: "+this.getHoursWorked();
    }
}

class Tester {

    public static void main(String[] args) {

        PermanentEmployee permanentEmployee = new 
PermanentEmployee(711211, "Rafael", 1850, 115, 3.5f);
        permanentEmployee.calculateMonthlySalary();
        System.out.println("Hi "+permanentEmployee.getEmployeeName()+", 
your salary is $"+Math.round(permanentEmployee.getSalary()*100)/100.0);

        ContractEmployee contractEmployee = new ContractEmployee(102, 
"Jennifer", 16, 90);
        contractEmployee.calculateSalary();
        System.out.println("Hi "+contractEmployee.getEmployeeName()+", 
your salary is $"+Math.round(contractEmployee.getSalary()*100)/100.0);
```

```
        //Create more objects for testing your code
    }

}
```

Method overloading 1

```
class Bill{
    //Implement your code here
    public double findPrice(int itemId) {
        if(itemId==1001) {
            return 25.0;
        }
        else if(itemId==1002) {
            return 20.0;
        }
        else if(itemId==1003) {
            return 23.0;
        }
        else if(itemId==1004) {
            return 18.0;
        }
        else return 0.0;
    }
    public double findPrice(String brandName, String itemType, int size)
{
        if(brandName=="Puma") {
            if(itemType=="T-shirt") {
                if(size==34 || size==36)
                return 25.0;
                else return 0.0;
            }
            else if(itemType=="Skirt") {
                if(size==38 || size==40)
                    return 20.0;
                else return 0.0;
            }
            else return 0.0;
        }
        else if(brandName=="Reebok") {
            if(itemType=="T-shirt") {
                if(size==34 || size==36)
                    return 23.0;
                else return 0.0;
            }
            else if(itemType=="Skirt") {
                if(size==38 || size==40)
                    return 18.0;
                else return 0.0;
            }
            else return 0.0;
        }
        else return 0.0;
    }
}

class Tester {

      public static void main(String[] args) {

            Bill bill = new Bill();

            double price = bill.findPrice(1001);
            if(price>0)
                System.out.println("Price of the selected item is 
$"+price);
            else
                System.out.println("The Item Id is invalid");

            price = bill.findPrice("Reebok","T-shirt",34);
            if(price>0)
                System.out.println("Price of the selected item is 
$"+price);
            else
                System.out.println("The values are not valid");
      }

}
```

Method overloading - 2

```
class Point{
    private double xCoordinate;
    private double yCoordinate;

    public Point(double xCoordinate, double yCoordinate) {
        this.xCoordinate = xCoordinate;
        this.yCoordinate = yCoordinate;
    }
    public double calculateDistance() {
        double result = Math.sqrt(xCoordinate*xCoordinate +
yCoordinate*yCoordinate);
        return(Math.round(result * 100.0) / 100.0);
    }
    public double calculateDistance(Point point) {
        double result = Math.sqrt(Math.pow(this.xCoordinate-
point.xCoordinate, 2.0) + Math.pow(this.yCoordinate-point.yCoordinate,
2.0));
        return(Math.round(result * 100.0) / 100.0);
    }
    public double getxCoordinate() {
        return xCoordinate;
    }

    public void setxCoordinate(double xCoordinate) {
        this.xCoordinate = xCoordinate;
    }

    public double getyCoordinate() {
        return yCoordinate;
    }

    public void setyCoordinate(double yCoordinate) {
        this.yCoordinate = yCoordinate;
    }

    //Uncomment the below method after implementation before verifying
    //DO NOT MODIFY THE METHOD

    public String toString(){
        return "Point\nxCoordinate:
"+this.getxCoordinate()+"\nyCoordinate: "+this.getyCoordinate();
    }
}

class Triangle {
    private Point point1;
    private Point point2;
    private Point point3;

    public Triangle() {
        this.point1 = new Point(0,0);
        this.point2 = new Point(1,1);
        this.point3 = new Point(2,5);
    }
    public Triangle(double point1XCoordinate, double point1YCoordinate,
double point2XCoordinate, double point2YCoordinate, double
point3XCoordinate, double point3YCoordinate)
    {
        this.point1 = new Point(point1XCoordinate, point1YCoordinate);
        this.point2 = new Point(point2XCoordinate, point2YCoordinate);
        this.point3 = new Point(point3XCoordinate, point3YCoordinate);
    }

    public Triangle(Point point1, Point point2, Point point3) {
        this.point1 = point1;
        this.point2 = point2;
        this.point3 = point3;
    }

    public Point getPoint1() {
        return point1;
    }

    public void setPoint1(Point point1) {
        this.point1 = point1;
    }

    public Point getPoint2() {
        return point2;
    }

    public void setPoint2(Point point2) {
        this.point2 = point2;
    }

    public Point getPoint3() {
        return point3;
    }

    public void setPoint3(Point point3) {
        this.point3 = point3;
    }

    public double calculatePerimeter() {
        double result = 0.0;
        result += point1.calculateDistance(point2);
        result += point2.calculateDistance(point3);
        result += point3.calculateDistance(point1);
        return(Math.round(result * 100.0) / 100.0);
    }
    public double calculateArea() {
        double result = 0.0;
        double a = point1.calculateDistance(point2);
        double b = point2.calculateDistance(point3);
        double c = point3.calculateDistance(point1);
        double s = (a+b+c)/2.0;
        result = Math.sqrt(s * (s-a) * (s-b) * (s-c));
        return(Math.round(result * 100.0) / 100.0);
    }
}



class Tester {

   public static void main(String[] args) {
         Triangle triangle1 = new Triangle();
         Triangle triangle2 = new Triangle(1, 2, 6, 5, 5, 1);

         Point point1 = new Point(2, 1);
         Point point2 = new Point(4, 4);
         Point point3 = new Point(9, 1);
         Triangle triangle3 = new Triangle(point1, point2, point3);


         System.out.println("Perimeter of triangle1 is
"+triangle1.calculatePerimeter());
         System.out.println("Area of triangle1 is
"+triangle1.calculateArea());

         System.out.println("Perimeter of triangle2 is
"+triangle2.calculatePerimeter());
         System.out.println("Area of triangle2 is
"+triangle2.calculateArea());

         System.out.println("Perimeter of triangle3 is
"+triangle3.calculatePerimeter());
         System.out.println("Area of triangle3 is
"+triangle3.calculateArea());

         //Create more objects of Triangle class for testing your code

   }
}
```

Method overriding -1

```
class User {
    private int id;
    private String userName;
    private String emailId;
    private double walletBalance;

    public User(int id, String userName, String emailId, double
walletBalance) {
        this.id = id;
        this.userName = userName;
        this.emailId = emailId;
        this.walletBalance = walletBalance;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getEmailId() {
        return emailId;
    }

    public void setEmailId(String emailId) {
        this.emailId = emailId;
    }

    public double getWalletBalance() {
        return walletBalance;
    }

    public void setWalletBalance(double walletBalance) {
        this.walletBalance = walletBalance;
    }

    public boolean makePayment(double billAmount) {
        if(billAmount <= this.walletBalance) {
            this.walletBalance -= billAmount;
            return true;
        }
        return false;
    }
}

class PremiumUser extends User {
    private int rewardPoints;

    public PremiumUser(int id, String userName, String emailId, double
walletBalance) {
        super(id, userName, emailId, walletBalance);
        this.rewardPoints = 0;
    }

    public int getRewardPoints() {
        return rewardPoints;
    }

    public void setRewardPoints(int rewardPoints) {
        this.rewardPoints = rewardPoints;
    }

    public boolean makePayment(double billAmount) {
        if(billAmount <= super.getWalletBalance()) {
            super.setWalletBalance(super.getWalletBalance() -
billAmount);
            this.rewardPoints += (int) (0.1 * billAmount);
            return true;
        }
        return false;
    }
}

class Tester {

    public static void main(String[] args) {

        User user = new User(101, "Joe", "joe@abc.com", 100);

            PremiumUser premiumUser = new PremiumUser(201, "Jill",
"jill@abc.com", 300);

            processPayment(user, 70);

            processPayment(premiumUser, 150);

            processPayment(premiumUser, 80);

            processPayment(premiumUser, 120);

    }

    public static void processPayment(User user, double billAmount) {
        if (user.makePayment(billAmount)) {
                  System.out.println("Congratulations " +
user.getUserName() + ", payment of $" + billAmount + " was
successful!");
            } else {
                  System.out.println("Sorry " + user.getUserName() + ",
you do not have enough balance to pay the bill!");
            }
            System.out.println("Your wallet balance is $" +
user.getWalletBalance());

            if (user instanceof PremiumUser) {
                  PremiumUser premiumUser = (PremiumUser) user;
                  System.out.println("You have " +
premiumUser.getRewardPoints() + " points!");
            }
            System.out.println();
    }
}
```

Abstract - 1

```
abstract class Student {
    private String studentName;
    private int[] testScores;
    private String testResult;

    public Student(String studentName) {
        this.studentName = studentName;
        testScores = new int[4];
    }
    public abstract void generateResult();
    public String getStudentName() {
        return studentName;
    }

    public void setStudentName(String studentName) {
        this.studentName = studentName;
    }

    public String getTestResult() {
        return this.testResult;
    }

    public void setTestResult(String testResult) {
        this.testResult = testResult;
    }

    public void setTestScore(int testNumber, int testScore) {
        this.testScores[testNumber] = testScore;
    }
    public int[] getTestScores() {
        return this.testScores;
    }
}

class UndergraduateStudent extends Student {
    public UndergraduateStudent(String studentName) {
        super(studentName);
    }
    public void generateResult() {
        int[] t = super.getTestScores();
        float avg = 0F;
        for(int i=0; i<t.length; i++) {
            avg += t[i];
        }
        avg = avg/t.length;
        //System.out.println("avg = "+avg);
        if(avg>=60) {
            super.setTestResult("Pass");
        }
        else {
            super.setTestResult("Fail");
        }
    }
}
```

```
class GraduateStudent extends Student {
    public GraduateStudent(String studentName) {
        super(studentName);
    }
    public void generateResult() {
        int[] t = super.getTestScores();
        float avg = 0F;
        for(int i=0; i<t.length; i++) {
            avg += t[i];
        }
        avg = avg/t.length;
        //System.out.println("avg = "+avg);
        if(avg>=70) {
            super.setTestResult("Pass");
        }
        else {
            super.setTestResult("Fail");
        }
    }
}

class Tester {

    public static void main(String[] args) {
        UndergraduateStudent undergraduateStudent = new
UndergraduateStudent("Philip");
        undergraduateStudent.setTestScore(0, 70);
        undergraduateStudent.setTestScore(1, 69);
        undergraduateStudent.setTestScore(2, 71);
        undergraduateStudent.setTestScore(3, 55);

        undergraduateStudent.generateResult();

        System.out.println("Student name:
"+undergraduateStudent.getStudentName());
        System.out.println("Result:
"+undergraduateStudent.getTestResult());

        System.out.println();

        GraduateStudent graduateStudent = new GraduateStudent("Jerry");
        graduateStudent.setTestScore(0, 70);
        graduateStudent.setTestScore(1, 69);
        graduateStudent.setTestScore(2, 71);
        graduateStudent.setTestScore(3, 55);
        graduateStudent.generateResult();

        System.out.println("Student name:
"+graduateStudent.getStudentName());
        System.out.println("Result : "+graduateStudent.getTestResult());

        //Create more objects of the classes for testing your code
    }
}
```

Method overriding - 1

```
class Faculty {
    private String name;
    private float basicSalary;
    private float bonusPercentage;
    private float carAllowancePercentage;

    public Faculty(String name, float basicSalary) {
        this.name = name;
        this.basicSalary = basicSalary;
        this.bonusPercentage = 4F;
        this.carAllowancePercentage = 2.5F;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public float getBasicSalary() {
        return basicSalary;
    }

    public void setBasicSalary(float basicSalary) {
        this.basicSalary = basicSalary;
    }

    public float getBonusPercentage() {
        return bonusPercentage;
    }
    public void setBonusPercentage(float bonusPercentage) {
        this.bonusPercentage = bonusPercentage;
    }

    public float getCarAllowancePercentage() {
        return carAllowancePercentage;
    }

    public void setCarAllowancePercentage(float carAllowancePercentage)
{
        this.carAllowancePercentage = carAllowancePercentage;
    }

    public double calculateSalary() {
        return(this.basicSalary +
this.basicSalary*carAllowancePercentage +
this.basicSalary*bonusPercentage);
    }
}

class OfficeStaff extends Faculty {
    private String designation;

    public OfficeStaff(String name, float basicSalary, String
designation) {
        super(name, basicSalary);
        this.designation = designation;
    }

    public String getDesignation() {
        return designation;
    }

    public void setDesignation(String designation) {
        this.designation = designation;
    }

    public double calculateSalary() {
        double res = super.getBasicSalary() *
(1+super.getBonusPercentage()+super.getCarAllowancePercentage());
        if(designation=="Accountant") {
            res += 10000.0F;
        }
        else if(designation=="Clerk") {
            res += 7000.0F;
        }
        else if(designation=="Peon") {
            res += 4500.0F;
        }
        return res;
    }
}

class Teacher extends Faculty {
    private String qualification;

    public Teacher(String name, float basicSalary, String qualification)
{
        super(name, basicSalary);
        this.qualification = qualification;
    }

    public String getQualification() {
        return qualification;
    }

    public void setQualification(String qualification) {
        this.qualification = qualification;
    }

    public double calculateSalary() {
        double res = super.getBasicSalary() *
(1+super.getBonusPercentage()+super.getCarAllowancePercentage());
        if(qualification=="Doctoral") {
            res += 20000.0;
        }
        else if(qualification=="Masters") {
            res += 18000.0F;
        }
        else if(qualification=="Bachelors") {
            res += 15500.0F;
        }
        else if(qualification=="Associate") {
            res += 10000.0F;
        }
        return res;
    }
}


class Tester {
      public static void main(String[] args) {
            Teacher teacher = new Teacher("Caroline", 30500f, "Masters");
            OfficeStaff officeStaff = new OfficeStaff("James", 24000f,
"Accountant");

            System.out.println("Teacher Details\n***************");
            System.out.println("Name: "+teacher.getName());
            System.out.println("Qualification:
"+teacher.getQualification());
            System.out.println("Total salary: $" +
Math.round(teacher.calculateSalary()*100)/100.0);
            System.out.println();

            System.out.println("Office Staff Details\n***************");
            System.out.println("Name: "+officeStaff.getName());
            System.out.println("Designation:
"+officeStaff.getDesignation());
            System.out.println("Total salary: $" +
Math.round(officeStaff.calculateSalary()*100)/100.0);

        //Create more objects for testing your code

      }
}
```

Method overriding - 2

```
class Event {
    private String eventName;
    private String participantName;
    private double registrationFee;

    public Event(String eventName, String participantName) {
        this.eventName = eventName;
        this.participantName = participantName;
    }

    public String getEventName() {
        return eventName;
    }

    public void setEventName(String eventName) {
        this.eventName = eventName;
    }
```

```java
    public String getParticipantName() {
        return participantName;
    }

    public void setParticipantName(String participantName) {
        this.participantName = participantName;
    }

    public double getRegistrationFee() {
        return registrationFee;
    }

    public void setRegistrationFee(double registrationFee) {
        this.registrationFee = registrationFee;
    }

    public void registerEvent() {
        if(eventName=="Singing") {
            this.registrationFee = 8;
        }
        else if(eventName=="Dancing") {
            this.registrationFee = 10;
        }
        else if(eventName=="DigitalArt") {
            this.registrationFee = 12;
        }
        else if(eventName=="Acting") {
            this.registrationFee = 15;
        }
        else this.registrationFee = 0;
    }
}

class SoloEvent extends Event {
    private int participantNo;

    public SoloEvent(String eventName, String participantName, int
participantNo) {
        super(eventName, participantName);
        this.participantNo = participantNo;
    }

    public int getParticipantNo() {
        return participantNo;
    }
    public void setParticipantNo(int participantNo) {
        this.participantNo = participantNo;
    }

    /*public void registerEvent() {

    }*/
}

class TeamEvent extends Event {
    private int noOfParticipants;
    private int teamNo;

    public TeamEvent(String eventName, String participantName, int
noOfParticipants, int teamNo) {
        super(eventName, participantName);
        this.noOfParticipants = noOfParticipants;
        this.teamNo = teamNo;
    }

    public int getNoOfParticipants() {
        return noOfParticipants;
    }

    public void setNoOfParticipants(int noOfParticipants) {
        this.noOfParticipants = noOfParticipants;
    }

    public int getTeamNo() {
        return teamNo;
    }

    public void setTeamNo(int teamNo) {
        this.teamNo = teamNo;
    }

    public void registerEvent() {
        String event = super.getEventName();
        if(event=="Singing") {
            super.setRegistrationFee(4*this.noOfParticipants);
        }
        else if(event=="Dancing") {
            super.setRegistrationFee(6*this.noOfParticipants);
        }
        else if(event=="DigitalArt") {
            super.setRegistrationFee(8*this.noOfParticipants);
        }
        else if(event=="Acting") {
            super.setRegistrationFee(10*this.noOfParticipants);
        }
        else {
            super.setRegistrationFee(0);
        }
    }
}

class Tester {

      public static void main(String[] args) {

        SoloEvent soloEvent = new SoloEvent("Dancing", "Jacob", 1);
            soloEvent.registerEvent();
            if (soloEvent.getRegistrationFee() != 0) {
                  System.out.println("Thank You " +
soloEvent.getParticipantName()
                              + " for your participation! Your
registration fee is $" + soloEvent.getRegistrationFee());
                  System.out.println("Your participant number is " +
soloEvent.getParticipantNo());

            } else {
                  System.out.println("Please enter a valid event");
            }

            System.out.println();
            TeamEvent teamEvent = new TeamEvent("Acting", "Serena", 5,
1);
            teamEvent.registerEvent();
            if (teamEvent.getRegistrationFee() != 0) {
                  System.out.println("Thank You " +
teamEvent.getParticipantName()
                              + " for your participation! Your
registration fee is $" + teamEvent.getRegistrationFee());
                  System.out.println("Your team number is " +
teamEvent.getTeamNo());
            } else {
                  System.out.println("Please enter a valid event");
            }
      }
}
```

Abstract 1

```
abstract class Payment {
    private int customerId;
    protected String paymentId;
    private double serviceTaxPercentage;

    public Payment(int customerId) {
        this.customerId = customerId;
    }

    public int getCustomerId() {
        return customerId;
    }

    public void setCustomerId(int customerId) {
        this.customerId = customerId;
    }

    public String getPaymentId() {
        return paymentId;
    }

    public void setPaymentId(String paymentId) {
        this.paymentId = paymentId;
    }

    public double getServiceTaxPercentage() {
        return serviceTaxPercentage;
    }

    public void setServiceTaxPercentage(double serviceTaxPercentage) {
        this.serviceTaxPercentage = serviceTaxPercentage;
    }

    public abstract double payBill(double amount);
}

class DebitCardPayment extends  Payment {
    private static int counter=1000;
    private double discountPercentage;

    public DebitCardPayment(int customerId) {
        super(customerId);
        super.setPaymentId("D"+counter);
    }
    public static int getCounter() {
        return counter;
    }

    public static void setCounter(int counter) {
        DebitCardPayment.counter = counter;
    }

    public double getDiscountPercentage() {
        return discountPercentage;
    }

    public void setDiscountPercentage(double discountPercentage) {
        this.discountPercentage = discountPercentage;
    }

    public double payBill(double amount) {
        if(amount <= 500) {
            super.setServiceTaxPercentage(2.5);
        }
        else if(amount>500 && amount<=1000) {
            super.setServiceTaxPercentage(4);
        }
        else { super.setServiceTaxPercentage(5); }

        if(amount <= 500) {
            this.discountPercentage = 1;
        }
        else if(amount>500 && amount<=1000) {
            this.discountPercentage = 2;
        }
        else { this.discountPercentage = 3; }
        counter++;
        return (amount *
(100+discountPercentage+super.getServiceTaxPercentage())/100F);
    }
}

class CreditCardPayment extends Payment {
    public static int counter = 1000;

    public CreditCardPayment(int customerId) {
        super(customerId);
        super.setPaymentId("C"+counter);
    }
    public static int getCounter() {
        return counter;
    }

    public static void setCounter(int counter) {
        CreditCardPayment.counter = counter;
    }

    public double payBill(double amount) {
        if(amount <= 500) {
            super.setServiceTaxPercentage(3);
        }
        else if(amount>500 && amount<=1000) {
            super.setServiceTaxPercentage(5);
        }
        else { super.setServiceTaxPercentage(6); }

        counter++;
        return(amount * (100+super.getServiceTaxPercentage())/100F);
    }
}

class Tester{
    public static void main(String args[]){
        DebitCardPayment debitCardPayment = new DebitCardPayment(101);
        double
billAmount=Math.round(debitCardPayment.payBill(500)*100)/100.0;
            System.out.println("Customer Id: " +
debitCardPayment.getCustomerId());
            System.out.println("Payment Id: " +
debitCardPayment.getPaymentId());
            System.out.println("Service tax percentage: " +
debitCardPayment.getServiceTaxPercentage());
            System.out.println("Discount percentage: " +
debitCardPayment.getDiscountPercentage());
            System.out.println("Total bill amount: " + billAmount);

            CreditCardPayment creditCardPayment = new
CreditCardPayment(102);

billAmount=Math.round(creditCardPayment.payBill(1000)*100)/100.0;
            System.out.println("Customer Id: " +
creditCardPayment.getCustomerId());
            System.out.println("Payment Id: " +
creditCardPayment.getPaymentId());
```

```
            System.out.println("Service tax percentage: " +
creditCardPayment.getServiceTaxPercentage());
            System.out.println("Total bill amount: " + billAmount);
    }
}
```

Final - 1

```
class Student {
    private final int STIPEND = 100;
    private int studentId;
    private int aggregateMarks;

    public int getStudentId() {
        return studentId;
    }

    public void setStudentId(int studentId) {
        this.studentId = studentId;
    }

    public int getAggregateMarks() {
        return aggregateMarks;
    }

    public void setAggregateMarks(int aggregateMarks) {
        this.aggregateMarks = aggregateMarks;
    }

    public int getSTIPEND() {
        return STIPEND;
    }

    public double calculateTotalStipend() {
        int bonus = 0;
        if(this.aggregateMarks >= 85 && this.aggregateMarks<90) {
            bonus = 10;
        }
        else if(this.aggregateMarks >= 90 && this.aggregateMarks<95) {
            bonus = 15;
        }
        else if(this.aggregateMarks >= 95 && this.aggregateMarks<=100) {
            bonus = 20;
        }
        return(STIPEND + bonus);
    }
}

class Tester {

      public static void main(String[] args) {
            Student student1 = new Student();
            student1.setStudentId(1212);
            student1.setAggregateMarks(93);

            double totalStipend = student1.calculateTotalStipend();
            System.out.println("The final stipend of " + 
student1.getStudentId()+" is $" + totalStipend);

            Student student2 = new Student();
            student2.setStudentId(1222);
            student2.setAggregateMarks(84);

            totalStipend = student2.calculateTotalStipend();
            System.out.println("The final stipend of " + 
student2.getStudentId()+" is $" + totalStipend);
      }

}
```

Interface - 1

```
interface Tax {
    abstract double calculateTax(double price);
}

class PurchaseDetails implements Tax {
    private String purchaseId;
    private String paymentType;
    private double taxPercentage;

    public PurchaseDetails(String purchaseId, String paymentType) {
        this.purchaseId = purchaseId;
        this.paymentType = paymentType;
    }

    public double getTaxPercentage() {
        return taxPercentage;
    }
    public void setTaxPercentage(double taxPercentage) {
        this.taxPercentage = taxPercentage;
    }

    public String getPurchaseId() {
        return purchaseId;
    }

    public void setPurchaseId(String purchaseId) {
        this.purchaseId = purchaseId;
    }

    public String getPaymentType() {
        return paymentType;
    }

    public void setPaymentType(String paymentType) {
        this.paymentType = paymentType;
    }

    public double calculateTax(double price) {
        if(paymentType=="Debit Card") {
            this.taxPercentage = 2;
        }
        else if(paymentType=="Credit Card") {
            this.taxPercentage = 3;
        }
        else {
            this.taxPercentage = 4;
        }
        return price*(1+taxPercentage);
    }
}

class Seller implements Tax {
    private String location;
    private double taxPercentage;

    public Seller(String location) {
        this.location = location;
    }

    public String getLocation() {
        return location;
    }
    public void setLocation(String location) {
        this.location = location;
    }

    public double getTaxPercentage() {
        return taxPercentage;
    }

    public void setTaxPercentage(double taxPercentage) {
        this.taxPercentage = taxPercentage;
    }

    public double calculateTax(double price) {
        if(location=="Middle east") {
            this.taxPercentage = 15;
        }
        else if(location=="Europe") {
            this.taxPercentage = 25;
        }
        else if(location=="Canada") {
            this.taxPercentage = 22;
        }
        else if(location=="Japan") {
            this.taxPercentage = 12;
        }
        else {
            this.taxPercentage = 0;
        }
        return price*(1+taxPercentage);
    }
}
```

Exception - 1

```
// Implement user defined exception classes
class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
    public String getMessage() {
        return(this.toString());
    }
}
class InvalidJobProfileException extends Exception {
    public InvalidJobProfileException(String message) {
        super(message);
    }
}
class InvalidNameException extends Exception {
    public InvalidNameException(String message) {
        super(message);
    }
}

class Applicant {

    private String name;
    private String jobProfile;
    private int age;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getJobProfile() {
         return jobProfile;
    }

    public void setJobProfile(String jobProfile) {
        this.jobProfile = jobProfile;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}


class Validator extends Applicant {
    public boolean validateName(String name) {
        if(name!=null) {
            return true;
        }
        return false;
    }
    public boolean validateJobProfile(String jobProfile) {
        if(jobProfile=="Clerk" || jobProfile=="Clerk" ||
jobProfile=="Executive" || jobProfile=="Officer") {
            return true;
        }
        return false;
    }
    public boolean validateAge(int age) {
        if(age>=18 && age<=30) {
            return true;
        }
        return false;
    }
    public boolean validate(Applicant applicant) throws
InvalidNameException, InvalidAgeException, InvalidJobProfileException {
        if(!validateName(applicant.getName())) {
            throw new InvalidNameException("Invalid name");
        }
        if(!validateAge(applicant.getAge())) {
            throw new InvalidAgeException("Invalid age");
        }
        if(!validateJobProfile(applicant.getJobProfile())) {
            throw new InvalidJobProfileException("Invalid Job Post");
        }
        return true;
    }
}

class Tester {

    public static void main(String[] args) {

        try {
            Applicant applicant= new Applicant();
            applicant.setName("Jenny");
            applicant.setJobProfile("Clerk");
            applicant.setAge(25);

            Validator validator = new Validator();

            validator.validate(applicant);
            System.out.println("Application submitted successfully!");
        }
        catch
```

```
(InvalidNameException|InvalidJobProfileException|InvalidAgeException e)
{
            System.out.println(e.getMessage());
        }
    }
}
```

Exception - assignment 1

```
class InvalidCouponCodeException extends Exception {
    public InvalidCouponCodeException(String message) {
        super(message);
    }
}
class InvalidDestinationException extends Exception {
    public InvalidDestinationException(String message) {
        super(message);
    }
}
class InvalidTripPackageException extends Exception {
    public InvalidTripPackageException(String message) {
        super(message);
    }
}

class BusBooking {
    private int bookingId;
    private String destination;
    private String tripPackage;
    private double totalAmount;

    public BusBooking(int bookingId, String destination, String
tripPackage) {
        this.bookingId = bookingId;
        this.destination = destination;
        this.tripPackage = tripPackage;
    }

    public int getBookingId() {
        return bookingId;
    }

    public void setBookingId(int bookingId) {
        this.bookingId = bookingId;
    }
    public String getDestination() {
        return destination;
    }

    public void setDestination(String destination) {
        this.destination = destination;
    }

    public String getTripPackage() {
        return tripPackage;
    }

    public void setTripPackage(String tripPackage) {
        this.tripPackage = tripPackage;
    }

    public double getTotalAmount() {
        return totalAmount;
    }

    public void setTotalAmount(double totalAmount) {
        this.totalAmount = totalAmount;
    }

    public boolean validateCouponCode(String couponCode, int 
numberOfMembers) {
        if(couponCode=="BIGBUS" && numberOfMembers>=10) {
            return true;
        }
        if(couponCode=="MAGICBUS" && numberOfMembers>=15) {
            return true;
        }
        else try {
            throw new InvalidCouponCodeException("Invalid Coupon");
        } catch (InvalidCouponCodeException e) {
            System.out.println(e.toString());
        }
        return false;
    }

    public String bookTrip(String couponCode, int numberOfMembers) {
        if(this.destination == "WashingtonDC" ||this.destination == 
"Philadelphia" ||
                this.destination == "Orlando" ||
                this.destination == "Boston" ||
this.destination=="Atlanta") {

            if(tripPackage=="Regular" || tripPackage=="Premium") {
                if(validateCouponCode(couponCode, numberOfMembers)) {
                    if(tripPackage=="Regular") {
                        this.totalAmount = 500;
                        return "Booking successful";
                    }
                    else {
                        this.totalAmount=800;
                        return "Booking successful";
                    }
                }
                else {
                    try {
                        throw new InvalidCouponCodeException("Invalid
Coupon");
                    } catch (InvalidCouponCodeException e) {
                        System.out.println(e.toString());
                    }
                }
            }
            else {
                try {
                    throw new InvalidTripPackageException("Invalid
package");
                } catch (InvalidTripPackageException e) {
                    System.out.println(e.toString());
                }
            }
        }
        else {
            try {
                throw new InvalidDestinationException("Invalid
Destination");
            } catch (InvalidDestinationException e) {
                System.out.println(e.toString());
            }
        }
        return null;
    }

}

class Tester{
    public static void main(String[] args) {
            BusBooking booking = new BusBooking(101,"Orlando",
"Regular");
            String result = booking.bookTrip("BIGBUS", 11);
            if(result.equals("Booking successful")){
                  System.out.println(result);
                  System.out.println("Total amount for the trip: " +
booking.getTotalAmount());
            }
            else{
                  System.out.println(result);
                  System.out.println("Your booking was not successful,
please try again!");
            }
      }
}
```

Final - assignment 1

```
class Circle {
    private final double PI=3.14;
    private double diameter;
    private double circumference;
    private double area;

    public Circle(double diameter) {
        this.diameter = diameter;
    }

    public double getPI() {
        return PI;
    }

    public double getDiameter() {
        return diameter;
    }

    public void setDiameter(double diameter) {
        this.diameter = diameter;
    }

    public double getCircumference() {
        return circumference;
    }
    public void setCircumference(double circumference) {
        this.circumference = circumference;
    }

    public double getArea() {
        return area;
    }

    public void setArea(double area) {
        this.area = area;
    }

    public void calculateCircumference() {
        this.circumference = PI * diameter;
    }
    public void calculateArea() {
        this.area = PI * diameter * diameter / 4F;
    }
}

class Tester{

    public static void main(String[] args) {

        Circle circle1 = new Circle(10.2);
        Circle circle2 = new Circle(5.7);

        //Create more objects of Circle class and add to the array given
below for testing your code
        Circle[] circles = {circle1, circle2};

        for (Circle circle : circles) {

            circle.calculateCircumference();
            circle.calculateArea();

            System.out.println("Diameter of the circle is
"+circle.getDiameter());
            System.out.println("Circumference of the circle is " +
Math.round(circle.getCircumference()*100)/100.0);
            System.out.println("Area of the circle is " +
Math.round(circle.getArea()*100)/100.0);
            System.out.println();
        }
    }
}
```

Interface 1

```
class Mobile {
    private String name;
    private String brand;
    private String operatingSystemName;
    private String operatingSystemVersion;

    public Mobile(String name, String brand, String operatingSystemName,
String operatingSystemVersion) {
        this.name = name;
        this.brand = brand;
        this.operatingSystemName = operatingSystemName;
        this.operatingSystemVersion = operatingSystemVersion;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getOperatingSystemName() {
        return operatingSystemName;
    }

    public void setOperatingSystemName(String operatingSystemName) {
        this.operatingSystemName = operatingSystemName;
    }

    public String getOperatingSystemVersion() {
        return operatingSystemVersion;
    }
    public void setOperatingSystemVersion(String operatingSystemVersion)
{
        this.operatingSystemVersion = operatingSystemVersion;
    }
}

interface Testable {
    public abstract boolean testCompatibility();
}

class SmartPhone extends Mobile implements Testable {
    private String networkGeneration;

    public SmartPhone(String name, String brand, String
operatingSystemName, String operatingSystemVersion, String
networkGeneration) {
        super(name, brand, operatingSystemName, operatingSystemVersion);
        this.networkGeneration = networkGeneration;
    }

    public String getNetworkGeneration() {
        return networkGeneration;
    }

    public void setNetworkGeneration(String networkGeneration) {
        this.networkGeneration = networkGeneration;
    }

    public boolean testCompatibility() {
        if(super.getOperatingSystemName()=="Saturn") {
            String t = super.getOperatingSystemVersion();
            if(networkGeneration=="3G") {
                if(t=="1.1" || t=="1.2" || t=="1.3") {
                    return true;
                }
            }
            else if(networkGeneration=="4G") {
                if(t=="1.2" || t=="1.3") {
                    return true;
                }
            }
            else if(networkGeneration=="5G") {
                if(t=="1.3") {
                    return true;
                }
            }
            else return false;
            return false;
        }
        else if(super.getOperatingSystemName()=="Gara") {
            String t = super.getOperatingSystemVersion();
            if(networkGeneration=="3G") {
                if(t=="EXTR.1" || t=="EXTR.2" || t=="EXTR.3") {
                    return true;
                }
            }
            else if(networkGeneration=="4G") {
                if(t=="EXTR.2" || t=="EXTR.1") {
                    return true;
                }
            }
            else if(networkGeneration=="5G") {
                if(t=="EXTR.1") {
                    return true;
                }
            }
            else return false;
            return false;
        }
        return false;
    }
}

class Tester {
      public static void main(String args[]){
            SmartPhone smartPhone = new SmartPhone("KrillinM20",
"Nebula", "Saturn", "1.3", "5G");
            if(smartPhone.testCompatibility())
                System.out.println("The mobile OS is compatible with the
network generation!");
            else
                System.out.println("The mobile OS is not compatible with
the network generation!");

            //Create more objects for testing your code
      }
}
```

Arraylist 1

```
import java.util.ArrayList;
import java.util.List;

class Order {
    private int orderId;
    private List<String> itemNames;
    private boolean cashOnDelivery;

    public Order(int orderId, List<String> itemNames, boolean
cashOnDelivery) {
        this.orderId = orderId;
        this.itemNames = itemNames;
        this.cashOnDelivery = cashOnDelivery;
    }

    public int getOrderId() {
        return orderId;
    }

    public void setOrderId(int orderId) {
        this.orderId = orderId;
    }

    public List<String> getItemNames() {
        return itemNames;
    }

    public void setItemNames(List<String> itemNames) {
        this.itemNames = itemNames;
    }

    public boolean isCashOnDelivery() {
        return cashOnDelivery;
    }

    public void setCashOnDelivery(boolean cashOnDelivery) {
        this.cashOnDelivery = cashOnDelivery;
    }

    @Override
    public String toString() {
        return "Order Id: "+getOrderId()+", Item names:
"+getItemNames()+", Cash on delivery: "+isCashOnDelivery();
    }
```

```java
}


class Tester {

    public static List<String> getItems(List<Order> orders) {
        //Implement your logic here and change the return statement
accordingly
        List<String> ret = new ArrayList<String>();
        for(Order order: orders) {
            List<String> t = order.getItemNames();
            for(String x: t) {
                ret.add(x);
            }
        }
        return ret;
    }

    public static void main(String[] args) {
        List<Order> orders = new ArrayList<Order>();

        List<String> items1 = new ArrayList<String>();
        items1.add("FriedRice");
        items1.add("Pasta");
        items1.add("Tortilla");
        orders.add(new Order(101, items1, true));

        List<String> items2 = new ArrayList<String>();
        items2.add("Pizza");
        items2.add("Pasta");
        orders.add(new Order(102, items2, true));

        List<String> items3 = new ArrayList<String>();
        items3.add("Burger");
        items3.add("Sandwich");
        items3.add("Pizza");
        orders.add(new Order(103, items3, true));

        List<String> items = getItems(orders);
        System.out.println("List of Items:");
        for (String item : items) {
            System.out.println(item);
        }

    }
```

```
 }
```

String-1

```
 class Tester {

       public static int findHighestOccurrence(String str){
             // Create array to keep the count of individual
         // characters and initialize the array as 0
         int count[] = new int[256];

         // Construct character count array from the input
         // string.
         int len = str.length();
         for (int i=0; i<len; i++)
             count[str.charAt(i)]++;

         int max = -1;  // Initialize max count
         char result = ' ';   // Initialize result

         // Traversing through the string and maintaining
         // the count of each character
         for (int i = 0; i < len; i++) {
             if (max < count[str.charAt(i)]) {
                 max = count[str.charAt(i)];
                 result = str.charAt(i);
             }
         }

         return result;
       }

       public static void main(String args[]){
           String str = "success";
           System.out.println(findHighestOccurrence(str));
       }
 }
```

String - 2

```
class Tester{
    public static String removeDuplicatesandSpaces(String str){
        // Used as index in the modified string
        int index = 0;
        int n = str.length();

        // Traverse through all characters
        for (int i = 0; i < n; i++)
        {

            // Check if str[i] is present before it
            int j;
            for (j = 0; j < i; j++)
            {
                if (str.charAt(i) == str.charAt(j))
                {
                    break;
                }
            }

            // If not present, then add it to
            // result.
            if (j == i)
            {
                str.charAt(index++) = str.charAt(i);
            }
        }
        return String.valueOf(String.copyOf(str, index));
      }

      public static void main(String args[]){
          String str = "object oriented programming";
          System.out.println(removeDuplicatesandSpaces(str));
      }
}

 class Tester{
     public static String removeDuplicatesandSpaces(String str){
         // Used as index in the modified string
         int index = 0;
         int n = str.length();

         // Traverse through all characters
         for (int i = 0; i < n; i++)
         {

             // Check if str[i] is present before it
             int j;
             for (j = 0; j < i; j++)
             {
                 if (str.charAt(i) == str.charAt(j))
                 {
                     break;
                 }
             }

             // If not present, then add it to
             // result.
             if (j == i)
             {
                 str.charAt(index++) = str.charAt(i);
             }
         }
         return String.valueOf(String.copyOf(str, index));
       }

       public static void main(String args[]){
           String str = "object oriented programming";
           System.out.println(removeDuplicatesandSpaces(str));
       }
 }
```

Array - 1

```
class Student{
    //Implement your code here
    private int[] marks;
    private char[] grades;

    public Student(int[] marks) {
        this.marks = marks;
        grades = new char[marks.length];
    }

    public void findGrade() {
        for(int i=0; i<marks.length; i++) {
            if(marks[i] >= 80) {
                grades[i] = 'S';
            }
            else if(marks[i] >= 60) {
                grades[i] = 'A';
            }
            else if(marks[i] >= 40) {
                grades[i] = 'B';
            }
            else if(marks[i] >= 20) {
                grades[i] = 'C';
            }
            else {
                grades[i] = 'F';
            }
        }
    }

    public char[] getGrade() {
        return grades;
    }
}

class Tester{
    public static void main(String[] args) {
        int[] marks = { 79, 87, 97, 65, 78, 99, 66 };
        Student student = new Student(marks);
        student.findGrade();
        System.out.println("Grades corresponding to the marks are : ");
        char[] grades = student.getGrade();
        for (int index = 0; index < grades.length; index++) {
            System.out.print(grades[index] + " ");
        }
    }
}
```

Array - 2

```
class Tester {
    public static int getCount(int[] numbers, int i) {
        int count=0;

        for(int j=i; j<numbers.length; j++) {
            if(numbers[i]==numbers[j]) count++;
        }

        return count;
    }

    public static int findTotalCount(int[] numbers) {
        //Implement your code here and change the return value
accordingly
```

```
        int count=0;

        for(int i=0; i<numbers.length-1; i++) {
            count += getCount(numbers, i);
        }

        return count;
    }

    public static void main(String[] args) {
        int[] numbers = { 1, 1, 5, 100, -20, 6, 0, 0 };
        System.out.println("Count of adjacent occurrence: 
"+findTotalCount(numbers));
    }
}
```