Java programming

DAY 1

```
1. class Tester {
       public static void main(String[] args) {
           System.out.println("My name is Sonakshi Gupta.\n My designation is System
Engineer.");
           // Implement your code here
       }
}

2. class Tester {
       public static void main(String[] args) {
           System.out.println("House no: 761 \n Ward No. 37 \n City: Jammu \n Pin code:
180007");
           // Implement your code here
       }
}
```

DAY 2

```
1. class Tester {
       public static void main(String[] args) {
           int principal=5000;
           float rate= 10.0f;
           byte time= 5;
           float interest= 0.0f;
           System.out.println((principal*rate*time)/100);
           // Implement your code here
       }
}

2. class Tester {
       public static void main(String[] args) {
           int principal=5000;
           float rate= 10.0f;
           byte time= 5;
           float interest= 0.0f;
           System.out.println((principal*rate*time)/100);
             int prin=3250;
           float r= 7.0f;
           byte t= 3;
           float Sinterest= 0.0f;
           System.out.println((prin*r*t)/100);

           // Implement your code here
       }
}
3. class Tester {
       public static void main(String[] args) {
           byte rad=4;
           float area=(float) (3.14*rad*rad);
           System.out.println(area);
             byte radius=10;
           float ar=(float) (3.14*radius*radius);
           System.out.println(ar);

           // Implement your code here
       }
}

4. class Tester {
        public static void main(String[] args) {
                float fahr= 32;
                float far= 50;
                float cels= (float)(((fahr-32)/9)*5);
                    float cel= (float)(((far-32)/9)*5);
                System.out.println(cels);
                System.out.println(cel);
        }
}
```

DAY 3

```
1. class Tester {
        public static void main(String[] args) {
                int num1=3;
                int num2=4;
                int num3=1;
                if(num1>num2)
              {
                if(num1>num3)
                {System.out.println(num1+ " is the maximum number");
                }
                else
                {System.out.println(num3+ " is the maximum number");
                }
                      }
                      else {
                        {System.out.println(num2+ " is the maximum number");
                }
                      }
        }
}
```

2. REVERSE OF A NUMBER

```
class Tester {
        public static void main(String[] args) {
                int inputNumber = 7865;
                int sumOfDigits = 0;
                int temp = 0;

                while (inputNumber > 0) {
                        temp = inputNumber % 10;
                        sumOfDigits = sumOfDigits*10+ temp;
                        inputNumber = inputNumber / 10;
                }

                System.out.println("Sum of digits are : " + sumOfDigits);
        }
}
```

3. SUM OF DIGITS IN A NUMBER|| USING WHILE LOOP

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

```
        int inputNumber = 7865;
        int sumOfDigits = 0;
        int temp = 0;

        while (inputNumber > 0) {
            temp = inputNumber % 10;
            sumOfDigits = sumOfDigits*10+ temp;
            inputNumber = inputNumber / 10;
        }

        System.out.println("Sum of digits are : " + sumOfDigits);
    }
}
```

4. INFINITE WHILE LOOP

```
class Tester {
    public static void main(String[] args) {
        int totalCost = 0;
        char wantToAddFoodItem = 'Y';
        int unitPrice = 10;
        int quantity = 1;

        while (wantToAddFoodItem == 'Y') {
            totalCost = totalCost + (quantity * unitPrice);
            System.out.println("Order placed successfully");
            System.out.println("Total cost: " + totalCost);
            System.out.println("Do you want to add one more food item to the order?");
        }
        System.out.println("Thank you for ordering the food! It will reach you shortly...");
    }
}
```

TREATING ININITE WHILE LOOP

```
class Tester {
    public static void main(String[] args) {
        int totalCost = 0;
        char wantToAddFoodItem = 'Y';
        int unitPrice = 10;
        int quantity = 1;

        while (wantToAddFoodItem == 'Y') {
            totalCost = totalCost + (quantity * unitPrice);
            System.out.println("Order placed successfully");
            System.out.println("Total cost: " + totalCost);
            System.out.println("Do you want to add one more food item to the order?");
            wantToAddFoodItem = 'N';
        }
        System.out.println("Thank you for ordering the food! It will reach you shortly...");
    }
}
```

5. SUM OF DIGITS IN A NUMBER|| USING DO-WHILE LOOP

```
class Tester {
    public static void main(String[] args) {
        int inputNumber = 9654;
        int sumOfDigits = 0;
        int temp = 0;
        do {
            temp = inputNumber % 10;
            sumOfDigits += temp;
            inputNumber = inputNumber / 10;
        } while (inputNumber > 0);

        System.out.println("Sum of digits : " + sumOfDigits);
    }
}
```
6. FOR LOOP EXECUTION

```
class Customer {
    public static void main(String[] args) {
        // The below code generates customerId
        int totalNoOfCustomers = 12;
        String customerId = "";
        for (int counter = 1; counter <= totalNoOfCustomers; counter++) {
            if (counter <= 9)
                customerId = "C0" + counter;
            else
                customerId = "C" + counter;
            System.out.println("Customer Id for customer " + counter + " is "
                    + customerId);
        }
    }
}
```

7. NESTED FOR LOOP|| PATTERN FORMATION

```
class Tester {
    public static void main(String[] args) {
        for (int row = 1; row <= 4; row++) {
            for (int value = 1; value <= 5; value++) {
                System.out.print(value + " ");
            }
            System.out.println();
        }
    }
}
```

```
class Tester {
    public static void main(String[] args) {
        for (int row = 1; row <= 4; row++) {
            for (int value = 1; value <= row; value++) {
                System.out.print(value + " ");
            }
            System.out.println();
        }
    }
}
```
----

8. FACTORIAL OF A NUMBER

```
class Tester {
    public static void main(String[] args) {
        int a=5;
        int temp=1;
        while(a!=0)
```

```
                {
                        temp= temp*a;
                        a--;
                }
                System.out.println(temp);
                // Implement your code here
        }
}
```

9. GEOMETRIC SEQUENCE

```
class Tester {
        public static void main(String[] args) {
                int a=1, r=2, n=5;
                int z=1;
                for(int i=1;i<=n;i++)
                {
                System.out.println(z);
                        int x= (int)Math.pow(r,i);
                z=a*x;


        }
}
}
```

SELECTION CONTROL STRUCTURE ASSIGNMENT

```
1. class Tester {
        public static void main(String[] args) {
                int a=5, b=5;
                int sum=0;
                if(a==b)
                {sum= a+b;
        }
        else{
                sum= 2*(a+b);
        }
        System.out.println(sum);
}
}
```

```
2. class Tester {
        public static void main(String[] args) {
        int a=1, b=4, c=6;
        float dis=0f;
        float x1=0f;
        float x2=0f;
        dis= (float)((b*b)-4*a*c);
        System.out.println(dis);
        if(dis==0)
        {
                System.out.println("the value of roots are equal: ");
        }
        else if(dis>0)
        {
                 System.out.println("the value of roots are unequal: ");
        }
        else{
                System.out.println("the value of roots are not real: ");
        }

          x1= (float)((-b+dis)/(2*a));
          x2= (float)((-b-dis)/(2*a));
          System.out.println(x1);
          System.out.println(x2);
        }
}
```

```
3. class Tester {
        public static void main(String[] args) {
                int a=2, b=6, c=7;
                int x=0;
                if(a==7)
                {
                        x= b*c;
                }
                else if(b==7)
                {
                        x=c;
                }
                else if(c==7)
                {
                        x=-1;
                }
                else
                {
                        x= a*b*c;
                }
                System.out.println(x);
        }
}
```

```
4. class Tester {
        public static void main(String[] args) {
        char foodType='N';
        int quan= 0;
        int dist= 1;
        int m=dist;
        int output= 0;
        int total=0;
        int extra=0;
        int extra1=1;
        int extra2=2;
        int cost=0;
          int charge=0;
          int sona=0;
        if(foodType=='N')
        {
                cost= 15;
        }
        else{
                cost= 12;
        }
        if (quan>= 1)
        {
                total= cost* quan;
        System.out.println("total cost is $" + total);
    }
    else{
        System.out.println("Invalid selection");
        output = -1;
        System.out.println("total bill amount is:" + output);
        System.out.println("order cannot be placed");
    }
    if(dist>=1)
    {
    if (dist<=3)
    {
        System.out.println("no delivery charges");
    }
    else if(dist<=6)
    {
    System.out.println("Additional delivery charges for $1 per km");
    extra= dist*extra1;
    }
    else if(dist<=m)
    {
    System.out.println("Additional delivery charges of $3 and $2 per km");
    extra= 3+ ((dist-6)*extra2);
    }
    output= total+ extra;
    System.out.println("your order has been successfully placed worth $" + output);
    }
    else
    {
        System.out.println("invalid selection " + " order cannot be placed");
    }

    }
}
```

```
5. class Tester {
    public static void main(String[] args) {
        int accnmbr=1001;
       int accbal= 250000;
       int salary= 40000;
       String loantype= "Car";
       int loanAmExpected= 300000;
       int emis= 30;
       int count=0;
       int eligibleloanAmount= 500000;
       int eligibleEmis= 36;
       int a= accnmbr;
       while(a!=0)
       {
          a=a/10;
            count++;
       }

      if(count==4)
      {
        int firstdigit=accnmbr/1000;
            if(firstdigit==1)
        {
            System.out.println("You can proceed further ");
```

```
                }
            }
            else{
                System.out.println("LOAN WILL NOT BE PROVIDED!!!");
            }
            if(accbal>=1000)
            {
                System.out.println("You are eligible for loan!");
            }
            else{
                System.out.println("LOAN WILL NOT BE PROVIDED!!!");
                }
             System.out.println("Checking whether the bank approves for loan");
             if(loanAmExpected<= eligibleloanAmount)
             {
                 if(emis<=eligibleEmis)
                 {
                     System.out.println("LOAN APPROVED!!!");
                     System.out.println("Account number: " + accnmbr);
         System.out.println("Loan type: " + loantype);
          System.out.println("Eligible loan amount: " + eligibleloanAmount);
          System.out.println("Loan amount requested: " + loanAmExpected);
          System.out.println("Number of eligible EMIs: " + eligibleEmis);
          System.out.println("Number of requested EMIs: " + emis);
                 }
             }
             else{
                 System.out.println("BANK CANNOT PROVIDE LOAN TO YOUR ACCOUNT!!!");
             }

         }
}
```

```
6.  class Tester {
        public static void main(String[] args) {
            int x=2,y=4,z=21;
            int n1=0;
            int n5=0;
            for(int i=1; i<=y; i++)
            {
                if(z>=5)
                {
                    n5=z/5;
                    z=z-(n5*5);
                }
            }

                for(int i=1; i<=x; i++)
            {
                if(z>=1)
                {
                    n1=z/1;
                    z=z-n1;
                }
            }
            if(n1>x)
            {
                System.out.println("-1");
            }
```

```
		else{
			System.out.print("$1 notes needed= ");
			System.out.println(n1);
		if(n5>y)
			{
				System.out.println("-1");
			}
			else{
					System.out.print("$5 notes needed= ");
				System.out.println(n5);
			}
		}
	}
}
```

```
7. class Tester {
	public static void main(String[] args) {
		int day=1;
		int month=9;
		int year=15;
		if(month==1|| month==3|| month==5|| month==7|| month==8|| month==10)
		{
		 if(day==31)
		 {
			day=1;
			month=month + 1;
		 }
		 else{
			 day=day+1;
		 }
		}
		else if(month==2)
		{
			if(((year%4==0) && (year%100!=0))    || (year%400==0))
			{
				System.out.println("Leap year");
				if(day==29)
				{
					day=1;
					month=month+1;
				}
				else{
					day= day+1;
				}
			}
			else{
				  if(day==28)
				  {
					  day=1;
					  month=month+1;
				  }
				  else{
					  day=day+1;
				  }
			  }
			}
			else if(month==4|| month==6|| month==9|| month==11)
			{
				if(day==30)
```

```
             {
                day=1;
                month=month + 1;
             }
             else{
                 day=day+1;
             }
                }
                else
                {
                    if(day==31)
                   {
                    day=1;
                    month=1;
                    year=year+1;
                }
                 else
                 {
                day=day+1;
             }

             }
                System.out.print(day + "-");
                System.out.print(month + "-");
                System.out.print("20" + year);
            }
       }
```

```
8. class Tester {
       public static void main(String[] args) {
              int n=3;
              if(n%3==0 && n%5==0)
              {
                     System.out.println("Zoom");
              }
              else if(n%5==0)
              {
                     System.out.println("Zap");
              }
              else if(n%3==0)
              {
                     System.out.println("Zip");
              }
              else
              {
                     System.out.println("Invalid");
              }
       }
}
```

ITERATION CONTROL STRUCTURE

```
1. class Tester {
       public static void main(String[] args) {
              int a=46763;
              int n=a;
```

```
            int rem=0;
            int quo=0;
            while(a!=0)
            {
                  rem=a%10;
                  quo=(quo*10)+rem;
               a=a/10;
            }
            System.out.println("Reverse of given number is: " + quo);
            if(quo==n)
            {
                  System.out.println(n + " is a pallindrome number");
            }
            else{
                     System.out.println(n + " is not a pallindrome number");
            }
      }
}
```

```
2. class Tester {
      public static void main(String[] args) {
      int heads=150, legs=500;
      int r=0, c=0;
      if((heads>legs)|| (heads==0) || (legs%2!=0))
      {
            System.out.println("The number of chickens and rabbits cannot be found");
      }
      else{
            r= (int)((legs+(-2*heads))/2);
            c= (int)(heads-r);
            System.out.println("Chickens= " + c);
            System.out.println("Rabbits= " + r);
      }
      }
}
```

```
3. class Tester {
      public static void main(String[] args) {
            int a=123, n=a, rem=0, temp=0, sum=0;
            while(n!=0)
            {
                  temp=n%10;
                  sum= sum+temp;
                  n=n/10;
            }
            System.out.println("sum of digits= " + sum);
            rem=a%sum;
            if(rem==0)
            {
                  System.out.println(a + " is divisible by sum of its digits");
            }
            else{
                  System.out.println(a + " is not divisible by sum of its digits");
            }
      }
}
```

```
4. class Tester {
      public static void main(String[] args) {
        int x=45, y=1000, rem=0, temp=0, rev=0, a=x;
        while(a!=0)
        {
            rem=a%10;
            a=a/10;
            temp=y*rem;
        }
        if(temp==y)
        {
            System.out.println(x + " is a seed of " + y);
        }
    else{
            System.out.println(x + " is not a seed of " + y);
    }
    }
}
----------------------
```

```
5. class Tester {
    public static void main(String[] args) {
        int a=1635;
        int n=a;
        int rem=0;
        int rev=0;
        double temp=0;
        int sum=0;
        while(a!=0)
        {
            rem=a%10;
            a=a/10;
            temp=Math.pow(rem, 3);
            sum=(int) (sum+ temp);
        }
        System.out.println(sum);
        if(sum==n)
        {System.out.println(n + " is an Armstrong number");
        }
        else{
          System.out.println(n + " is not an armstrong number");
        }
    }
}
---------------
```

```
6. class Tester {
    public static void main(String[] args) {
        String a="1623";
        int sum=0;
        for(int i=1; i<= a.length(); i=i+2)
        {
            int temp= a.charAt(i)-'0';
            sum=sum+(temp*temp);
        }
                if(sum%9==0)
                {
                    System.out.println(a + " is a lucky number");
                }
                else
                {
                    System.out.println(a + " is not a lucky number");
                }
        }
}
--------------------------
7. class Tester {
        public static void main(String[] args) {
                int num1=7, num2=9, lcm=0, max=0;
                if(num1>num2)
                max=num1;
                else
                max=num2;
                int step=max;
                while(num1!=0)
                {
                        if(max%num1==0 && max%num2==0)
                        {
                                lcm=max;
                                break;
                        }
                max=max+step;
                }
                System.out.println(lcm);
        }
}

--------------------
8. class Tester {
        public static void main(String[] args) {
                int n=5;
                for(int i=1; i<=n;i++)
                {
                        for(int j=n;j>=i;j--)
                        {
                                System.out.print("*" +" " );
                        }
                      System.out.println();
                }
        }
}
----------------------
-------------------------------------------------------------------
DAY 4
----------------------
1. CLASS INTRODUCTION TRYOUT
class Customer {

        public String customerId;
        public String customerName;
        public long contactNumber;
        public String address;

        public void displayCustomerDetails() {
                System.out.println("Displaying customer details \n***************************");
                System.out.println("Customer Id : " + customerId);
                System.out.println("Customer Name : " + customerName);
                System.out.println("Contact Number : " + contactNumber);
                System.out.println("Address : " + address);
                System.out.println();
        }

}
```

```
class Tester {

    public static void main(String[] args) {

        // Object creation
        Customer customer = new Customer();

        // Assigning values to the instance variables
        customer.customerId = "C101";
        customer.customerName = "Stephen Abram";
        customer.contactNumber = 7856341287L;
        customer.address = "D089, St. Louis Street, Springfield, 62729";

        // Displaying the customer details
        customer.displayCustomerDetails();
        // Move the above statement immediately after the object creation
        // statement and observe the output

    }

}
```
----

2. CLASS AND OBJECT-EXERCISE 1

```
class Customer {

    public String customerId;
    public String customerName;
    public long contactNumber;
    public String address;

    public void displayCustomerDetails() {
        System.out.println("Displaying customer details \n***************************");
        System.out.println("Customer Id : " + customerId);
        System.out.println("Customer Name : " + customerName);
        System.out.println("Contact Number : " + contactNumber);
        System.out.println("Address : " + address);
        System.out.println();
    }

}

class Tester {

    public static void main(String[] args) {

        // Object creation
        Customer customer = new Customer();

        // Assigning values to the instance variables
        customer.customerId = "C101";
        customer.customerName = "Stephen Abram";
        customer.contactNumber = 7856341287L;
        customer.address = "D089, St. Louis Street, Springfield, 62729";

        // Displaying the customer details
        customer.displayCustomerDetails();
        // Move the above statement immediately after the object creation
        // statement and observe the output
    }

}
----------------------------
```
3. LOCAL VARIABLE-TRYOUT
```
class Demo {
    public int var1; // Instance variable of the class

    public void printValue() {
        int var2 = 20; // Local variable of the method
        System.out.println(var1);
        System.out.println(var2); // Local variable is accessible only inside
                                          // the method
    }
}

class Tester {

    public static void main(String args[]) {
        Demo demo = new Demo();
        demo.var1 = 10; // Instance variable can be accessed from outside the
                          // class with the help of object
        demo.printValue();

        // Local variables cannot be accessed outside a method
        // Below lines will lead to a compilation error saying that var2 is not
        // a field or variable
        // System.out.println(demo.var2);
        // System.out.println(var2);
    }

}
----------------------
```
4. *METHODS- EXERCISE 1*
```
class Calculator {
    public double findAverage(int number1, int number2, int number3)
    {

        double sum= (number1+number2+number3);
         double average= sum/3;
         double roundoff= Math.round(average*100.0)/100.0;
        return roundoff;

    }

}

class Tester {

    public static void main(String args[]) {
        Calculator calculator = new Calculator();
        double x=calculator.findAverage(12,8,15);
        System.out.println(x);

        // Invoke the method findAverage of the Calculator class and display the average
    }
}
--------------------------
```
DAY 5

```
----
```

1. CONSTRUCTOR AND THIS KEYWORD: EXERCISE 1

```
class Customer {

	public String customerId;
	public String customerName;
	public long contactNumber;
	public String address;


	Customer(String customerName, long contactNumber, String address)
	{
		customerId= customerId;
		this.customerName= customerName;
		this.contactNumber= contactNumber;
		this.address= address;
	}
		public void displayCustomerDetails() {
		System.out.println("Displaying customer details");
		System.out.println("Customer Id : " + customerId);
		System.out.println("Customer Name : " + customerName);
		System.out.println("Contact Number : " + contactNumber);
		System.out.println("Address : " + address);
		System.out.println();
	}
}

class Tester {

	public static void main(String[] args) {

		Customer customer = new Customer("Jacob", 627295480, "St. Louis, New York");


		customer.displayCustomerDetails();

	}

}
```

METHODS-ASSIGNMENT

```
1. class Order{
	public int orderId;
	public String orderedFood;
	public double totalPrice;
	public String status;

	public double calculateTotalPrice(int unitPrice)
	{
		System.out.println("Order Details");
		totalPrice= unitPrice* (1+(5/100.0));
		return totalPrice;
	}
	public void displayDetails()
	{
		System.out.println("Order Id: "+ orderId);
		System.out.println("Ordered Food: " + orderedFood);
		System.out.println("Order Status: " + status);

	}
```

```
}
class Tester{
        public static void main(String args[]){
                Order o= new Order();
                o.orderId= 101;
                o.orderedFood= "Pasta";
                o.status= "ordered";
                double x= o.calculateTotalPrice(100);

                o.displayDetails();
                  System.out.println("Total Price: " + x);
        }
}
-----------------------
2. class Restaurant{
        public float rating;
        public String restaurantName;
        public long restaurantContact;
        public String restaurantAddress;


        public void displayRestaurantDetails()
        {
                System.out.println("Restaurant Details\n************************");
                System.out.println("Restaurant Name: "+ restaurantName);
                System.out.println("Restaurant Contact: " + restaurantContact);
                System.out.println("Restaurant Address: " + restaurantAddress);
                System.out.println("Rating: " + rating);
        }
}
class Tester{
        public static void main(String args[]){
                Restaurant r= new Restaurant();
                r.restaurantName= "Dominos";
                r.restaurantContact= 2534512;
                r.restaurantAddress= "Janipur colony";
                r.rating= 4.3f;
                r.displayRestaurantDetails();
        }
}
-------------------------------
3. class Calculator {
public int num;
public int sumOfDigits()
{
        int a=num,rem=0, sum=0;
        while(a!=0)
        {
                rem=a%10;
                a=a/10;
                sum=sum+rem;
        }
        return sum;
}

}

class Tester {

        public static void main(String args[]) {
            Calculator calculator = new Calculator();
            calculator.num= 123;
            int x= calculator.sumOfDigits();
            System.out.println(x);

            // Assign a value to the member variable num of Calculator class

            // Invoke the method sumOfDigits of Calculator class and display the output

        }
}
------------------------
4. class Rectangle {
        public float length;
        public float width;
        public double calculateArea()
        {
            double a= length* width;
            double area= Math.round(a*100.0)/100.0;
            return area;
        }
        public double calculatePerimeter()
        {
          double p= 2* (length+ width);
          double perimeter= Math.round(p*100.0)/100.0;
          return perimeter;
        }
}

class Tester {

        public static void main(String args[]) {

            Rectangle rectangle=new Rectangle();
            //Assign values to the member variables of Rectangle class
            rectangle.length=12f;
            rectangle.width= 5f;

            //Invoke the methods of the Rectangle class to calculate the area and perimeter
            double x= rectangle.calculateArea();
            double y= rectangle.calculatePerimeter();

            //Display the area and perimeter using the lines given below
            System.out.println("Area of the rectangle is " + x);
            System.out.println("Perimeter of the rectangle is " + y);
        }

}
--------------------
```

CONSTRUCTORS AND THIS KEYWORD

```
1. class Order{
        public int orderId;
        public String orderedFood;
        public double totalPrice;
        public String status;


        public Order()
        {
                status= "ordered";
        }
        public Order(int orderId, String orderedFood)
        {
                this.orderId= orderId;
                this.orderedFood= orderedFood;
                status="Ordered";
        }
}
class Tester{
        public static void main(String args[]){
                Order o1= new Order();
           System.out.println("Status of order 1: " + o1.status);
           Order o2= new Order(1001, "Pizza");
           System.out.println("Status of order 2: " + o2.status);
           System.out.println("Order Id of order 2: " + o2.orderId);
           System.out.println("Ordered Food for order 2: " + o2.orderedFood);

        }
}
----------------------
2.   class Restaurant{
        public float rating;
        public String restaurantName;
        public long restaurantContact;
        public String restaurantAddress;


        public void displayRestaurantDetails()
        {
                System.out.println("Restaurant Details\n************************");
                System.out.println("Restaurant Name: "+ restaurantName);
                System.out.println("Restaurant Contact: " + restaurantContact);
                System.out.println("Restaurant Address: " + restaurantAddress);
                System.out.println("Rating: " + rating);
        }

public Restaurant(String name, long restaurantContact, String restaurantAddress, float rating)
{
        restaurantName= name;
        this.restaurantContact= restaurantContact;
        this.restaurantAddress= restaurantAddress;
        this.rating= rating;
}
}
class Tester{
        public static void main(String args[]){
                Restaurant r= new Restaurant("Dominos",2534512,"Janipur colony", 4.3f    );
                r.displayRestaurantDetails();
        }
}
-------------------------
```
ENCAPSULATION-EXREСISE 1

```
class Employee {

        private String employeeId;
        private String employeeName;
        private int salary;
```

```
	private int bonus;
	private int jobLevel;

	public String getEmployeeId(){
		return employeeId;
	}
	public void setEmployeeId(String employeeId){
		this.employeeId= employeeId;
	}

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

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

	public int getBonus(){
		return bonus;
	}
	public void setBonus(int bonus){
		this.bonus= bonus;
	}

	public int getJobLevel(){
		return jobLevel;
	}
	public void setJobLevel(int jobLevel){
		this.jobLevel= jobLevel;
	}

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

class Tester {

	public static void main(String args[]) {

		Employee employee = new Employee();
		employee.setEmployeeId("C101");
		employee.setEmployeeName("Steve");
		employee.setSalary(650);
		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());

        }
}
```

ENCAPSULATION ASSIGNMENT

```
1.   class Order{
        private int orderId;
        private String orderedFood;
        private double totalPrice;
        private String status;

        public void setOrderedFood(String orderedFood){
            this.orderedFood= orderedFood;
        }

        public String getOrderedFood(){
            return orderedFood;
        }
        public void setStatus(String status){
            this.status= status;
        }

        public String getStatus(){
            return status;
        }
        public void setOrderId(int orderId){
            this.orderId= orderId;
        }

        public int getOrderId(){
            return orderId;
        }


        public double calculateTotalPrice(int unitPrice)
        {
            System.out.println("Order Details");
            totalPrice= unitPrice* (1+(5/100.0));
            return totalPrice;
        }

}
class Tester{
        public static void main(String args[]){
            Order o= new Order();
            o.setOrderId(101);
            o.setOrderedFood("Pasta");
            o.setStatus("ordered");

           double x= o.calculateTotalPrice(100);

             System.out.println("Order Id: "+ o.getOrderId());
            System.out.println("Ordered Food: " + o.getOrderedFood());
            System.out.println("Order Status: " + o.getStatus());
          System.out.println("Total amount to be paid: " + x);
        }
}
---------------------
2. class MovieTicket {
      private int movieId;
      private int noOfSeats;
      private double costPerTicket;

        public void setMovieId(int movieId){
                this.movieId= movieId;
        }

        public int getMovieId(){
                return movieId;
        }
        public void setNoOfSeats(int noOfSeats){
                this.noOfSeats= noOfSeats;
        }

        public int getNoOfSeats(){
                return noOfSeats;
        }
        public void setCostPerTicket(double costPerTicket){
                this.costPerTicket= costPerTicket;
        }

        public double getCostPerTicket(){
                return costPerTicket;
        }

      public MovieTicket(int movieId, int noOfSeats)
      {
              this.movieId= movieId;
              this.noOfSeats= noOfSeats;
      }
      public double calculateTotalAmount()
      {
              double a;

              switch(movieId)
              {
                      case 111: costPerTicket= 7;
                      break;
                      case 112: costPerTicket=8;
                      break;
                      case 113: costPerTicket=8.5;
                      break;
                      default: costPerTicket=0;
              }
            a= costPerTicket*noOfSeats;
              double amount= a*(1+(2/100.0));
              return amount;
      }
}

class Tester {
        public static void main(String[] args) {
                MovieTicket movieTicket = new MovieTicket(112, 3);
                movieTicket.setCostPerTicket(7);
                double amount = movieTicket.calculateTotalAmount();
        if (amount==0)
            System.out.println("Sorry! Please enter valid movie Id and number of seats");
        else
            System.out.println("Total      amount      for      booking      :      $"      +
Math.round(amount*100)/100.0);
    }
}
```

STRING METHODS-TRYOUT

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

        // length()
        String str = "Welcome";
        System.out.println(str.length());

        // concat()
        String s = "Hello";
        s.concat(" World");
        System.out.println(s);
        // s is still "Hello"
        // String objects are immutable which means they cannot be changed
        // Here, when we concat the two strings a new string object gets created

        String s1 = s.concat("World");
        System.out.println(s1);

        // + operator can also be used for string concatenation
        String fname = "Jack";
        String lname = "Black";
        System.out.println(fname + " " + lname);

        // equals()
        System.out.println(s.equals("Hello"));

        // equals compares only the values of the strings
        String s2 = new String("Hello");
        System.out.println(s.equals(s2));

        // == compares the object reference and will return false in the below
        // case
        System.out.println(s == s2);

        // equalsIgnoreCase()
        System.out.println(s.equalsIgnoreCase("hello"));

        // toLowerCase() and toUpperCase()
        System.out.println(str.toLowerCase());
        System.out.println(str.toUpperCase());

        // substring()
        String subs = "Learning is fun";
        System.out.println(subs.substring(4, 8));
        System.out.println(subs.substring(4));

        // charAt()
        System.out.println(subs.charAt(10));

        // contains()
        System.out.println(subs.contains("is"));
```

```
            // replace()
            System.out.println(subs.replace('i', 'k'));
        }

}
```

STRING EXERCISE

```
class Tester{

        public static String removeWhiteSpaces(String str){
               String noSpaceStr = str.replaceAll("\\s", ""); // using built in method
            //Implement your code here and change the return value accordingly
            return noSpaceStr;
        }

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

DAY 6

MULTIDIMENSIONAL ARRAY-TRYOUT

```
//Program to illustrate the use of multidimensional array
class Tester {
        public static void main(String[] args) {
            // Declaring and initializing 2D array
            int[][] dayWiseTemperature = new int[][] { { 29, 21 }, { 24, 23 },
                    { 26, 22 }, { 28, 23 }, { 29, 24 }, { 23, 20 }, { 29, 21 } };

            // Displaying 2D array
            for (int i = 0; i < 7; i++) {
                for (int j = 0; j < 2; j++) {
                    if (j == 0)
                        System.out.println("Max Temperature is "
                                + dayWiseTemperature[i][j] + " on day " + i);
                    else
                        System.out.println("Min Temperature is "
                                + dayWiseTemperature[i][j] + " on day " + i);
                }
            }
        }
}
```

ARRAY-EXERCISE 1

```
class Tester {

        public static int calculateSumOfEvenNumbers(int[] numbers){
            int sum=0;
            for(int i=0; i<numbers.length; i++)
            {
                int a=numbers[i]%2;
                if (a==0)
                {
                    sum=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 ASSIGNMENT

```
1. class Tester{

        public static String moveSpecialCharacters(String str){
               String regex= "[^a-zA-Z0-9\\s+]";
               String inputData= "";
               String specialCharacters= "";
               for(int i=0; i< str.length(); i++)
               {
                      char ch= str.charAt(i);
                      if(String.valueOf(ch).matches(regex))
                      {
                             specialCharacters= specialCharacters+ch;
                      }
                      else{
                             inputData= inputData+ch;
                      }
               }
               String result= inputData+specialCharacters;
               return result;
        }

        public static void main(String args[]){
               String str = "He@#$llo!*&";
               System.out.println(moveSpecialCharacters(str));
        }

}
```

```
2. class Tester{
        public static boolean checkPalindrome(String str){
               int length= str.length();
               String rev= "";

               for(int i=length-1; i>=0; i--)
               {

                      rev= rev + str.charAt(i);
               }
                      if (str.equals(rev))
                      {
                      return true;
}
  return false;
        }

        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!");
     }
}
----------------
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+ " ";
          }
          return reversedString.trim();
     }

     public static void main(String args[]){
          String str = "I love programming";
          System.out.println(reverseEachWord(str));
     }
}
-----------------
4. class Tester {
public static int findHighestOccurrence(String str){
          int counter=0;
          int max_counter=0;
          char ch;
          for(int i=0;i<str.length();i++)
          {
               ch=str.charAt(i);
               counter=0;
               for(int j=0;j<str.length();j++)
               {
                    if(ch==str.charAt(j))
                         counter++;
               }
               if(counter>max_counter)
               {
                    max_counter=counter;
               }
          }
          return max_counter;
     }

     public static void main(String args[]){
          String str = "success";
          System.out.println(findHighestOccurrence(str));
     }
}
------------------------
5. class Tester{
     public static String removeDuplicatesandSpaces(String str){
            String result = "";
        for (int i = 0; i < str.length(); i++) {
                if(!result.contains(String.valueOf(str.charAt(i)))) {
                        result += String.valueOf(str.charAt(i));
                }
        }
        return result.replace(" ","");

        }

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


ARRAY-ASSIGNMENTS

```
1. class Teacher {
        private String teacherName;
        private String subject;
        private double salary;
        public void setTeacherName(String teacherName){
                this.teacherName= teacherName;
        }

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

        public String getSubject(){
                return subject;
        }
        public void setSalary(double salary){
                this.salary= salary;
        }

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

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

                Teacher[] teacherobjects= new Teacher[4];
                teacherobjects[0]= new Teacher("Alex, ","Java Fundamental, ",1200.0);
                teacherobjects[1]= new Teacher("John, ","RDBMS, ", 800.0 );
                teacherobjects[2]= new Teacher("Sam, ","Networking, ",900.0 );
                teacherobjects[3]= new Teacher("Maria, ","Python, ",900.0 );
                for(int i=0; i< teacherobjects.length; i++)
                {
                    System.out.print("Name: "+teacherobjects[i].getTeacherName()+ " ");
                    System.out.print("Subject: "+teacherobjects[i].getSubject()+" ");
                    System.out.println("Salary: "+teacherobjects[i].getSalary()+" ");

                }
        }
}
```

```
2. class Tester {

    public static double[] findDetails(double[] salary) {
        //Implement your code here and change the return value accordingly
        double[] arr = new double[3];
        double avg = 0;
        for(int i = 0; i < salary.length; i++){
            avg += salary[i];
            if(i == salary.length -1) avg /= (double)(i+1);
        }
        arr[0] = avg;
        double greater = 0, lesser = 0;
        for(double data: salary){
            if(data > avg) greater++;
            if(data < avg) lesser++;
        }
        arr[1] = greater;
        arr[2] = lesser;
        return arr;

    }

    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]);
    }
}
```

```
3.   class Tester {

    public static int[] findLeapYears(int year){
        //Implement your code here and change the return value accordingly
        int[] ans = new int[15];
        int first = year;
        while(true){
            if(first % 4 == 0){
                if(first % 100 == 0){
                    if(first % 400 == 0) break;
                }else break;
            }
            first++;
        }
        int a = 0;
        for(int i = first; a < ans.length; i+=4){
            if(i % 100 == 0 && i % 400 != 0) continue;
```

```
                ans[a++] = i;
            }

              return ans;
    }

    public static void main(String[] args) {
            int year = 1684;
            int[] leapYears;
            leapYears=findLeapYears(year);
            for ( int index = 0; index<leapYears.length; index++ ) {
                System.out.println(leapYears[index]);
            }
       }
}
--------------------------------
4. class Student{
        private int[] marks;
        private char[] grade;

        public int[] getMarks() {
return marks;
}

public void setMarks(int[] marks) {
this.marks = marks;
}

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

public void setGrade(char[] grade) {
this.grade = grade;
}

public Student(int[] marks)
        {
        this.marks= marks;
        grade = new char[marks.length];
        }
public void findGrade()
{
for(int i=0; i< marks.length; i++)
{
if(marks[i]>=92)
{
grade[i]= 'E';
}
else if(marks[i]>=85 && marks[i]<92)
{
grade[i]= 'A';
}
else if(marks[i]>=70 && marks[i]<85)
{
grade[i]= 'B';
}
else if(marks[i]>=65 && marks[i]<70)
{
grade[i]= 'C';
}
else
{
grade[i]= 'D';
}
}
}
}

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] + " ");
}
}
}
```

```
5. class Tester {

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

		// Implement your code here
		if(num1 > num2) return numbers;
		if(num1 >= 100 || num2 < 10) return numbers;

		int a = 0;
		for(int i = num1 >= 10? num1+1 : 10; i <= num2 && i < 100; i++){
			if(i % 5 == 0 && i % 3 == 0) numbers[a++] = i;
		}

		return numbers;
	}

	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]);
			}
		}

	}
}
```

```
6. class Tester {
```

```
public static int findTotalCount(int[] numbers) {
int length= numbers.length;
int count=0;
for(int i=0; i<length-1; i++)
{

    if(numbers[i]==numbers[i+1])
    {
         count= count+1;
    }
}
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));
}
}
```

```
7. class Tester{
        public static String[] findPermutations(String str){
                //Implement your code here and change the return value accordingly

                String[] ans = new String[6];
                int a = 0;
                for(int i = 0; i < str.length(); i++){
                        String x = ""+str.charAt(i);
                        for(int j = 0; j < str.length(); j++){
                                if(j != i){
                                        String y = x + str.charAt(j);
                                        for(int k = 0; k < str.length(); k++){
                                                if(k != i && k != j){
                                                        String z = y + str.charAt(k);
                                                        ans[a++] = z;
                                                }
                                        }
                                }
                        }
                }
                int duplicates = 0;
                for(int i = 0; i < ans.length; i++){
                        String temp = ans[i];
                        if(!temp.equals("")){
                                for(int j = i+1; j < ans.length; j++){
                                        if(temp.equals(ans[j])){
                                                ans[j] = "";
                                                duplicates++;
                                                i-=1;
                                                break;
                                        }
                                }
                        }
                }
                String[] final_ans = new String[6];
                int fill = 0;
                for(int i = 0; i < ans.length; i++){
                        if(!ans[i].equals("")) final_ans[fill++] = ans[i];
                }
        return final_ans;
    }

    public static void main(String args[]){
        String str = "abc";
        String permutations[] = findPermutations(str);
        for(String permutation: permutations){
            if (permutation!=null)
                System.out.println(permutation);
        }
    }
}
----------------------
WEEK 2
-------
DAY 1
-----------
1. STATIC- TRYOUT
class Tester {
    public static void main(String[] args) {
        Car c1 = new Car("Red");
        Car c2 = new Car("Green");
        Car c3 = new Car("Blue");
        System.out.println("Number of cars created: " + Car.getNumberOfCars());
    }

    static {
        System.out.println("Tester class loaded");
    }
}

class Car {
    private static String color;
    private static int numberOfCars = 0;

    static {
        System.out.println("Car class loaded");
    }

    public Car(String color) {
        this.color = color;
        numberOfCars++;
    }

    public static String getColor() {
        return color;
    }

    public static int getNumberOfCars() {
        return numberOfCars;
    }
}
-------------
*EXERCISE*
class Bill{
private static int counter;
private String billId;
private String paymentMode;
```

```
static{
	counter=9000;
}
public Bill(String paymentMode){
this.paymentMode= paymentMode;
billId= "B" + ++counter;
}

public static int getCounter() {
return counter;
}

public String getBillId() {
return billId;
}
public void setBillId(String billId) {
this.billId = billId;
}
public String getPaymentMode() {
return paymentMode;
}
public void setPaymentMode(String paymentMode) {
this.paymentMode = paymentMode;
}

}

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

		Bill bill1 = new Bill("DebitCard");
		Bill bill2 = new Bill("PayPal");
		Bill bill3 = new Bill("CreditCard");
		Bill bill4 = new Bill("PayTm");
		Bill bill5 = new Bill("GooglePay");

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

		Bill[] bills = { bill1, bill2, bill3, bill4, bill5 };

		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();
		}
	}
}
```

AGGREGATION AND ACCESS MODIFIERS- TRYOUT

```
class Subject {
	private String subjectName;
	public void setSubjectName(String subjectName){
		this.subjectName= subjectName;
	}

	public String getSubjectName(){
		return subjectName;
	}
	Subject(String subjectName) {
		this.subjectName = subjectName;
	}
}

class Student {
	private int rollNo;
	private String studentName;
	private Subject subject;

	Student(int rollNo, String studentName, Subject subject) {
		this.rollNo = rollNo;
		this.studentName = studentName;
		this.subject = subject;
	}

	public void displayDetails() {
		System.out.println("Student Name: " + studentName);
		System.out.println("Subject Name: " + subject.getSubjectName());

		// We cannot directly access the private member of the class Subject
		// To access the private members of aggregated class, we will have to
		// make use of the getter and setter methods

		// Add the getter and setter methods to class Subject and modify the
		// displayDetails method accordingly

	}

	public static void main(String args[]) {
		Subject subject = new Subject("Maths");
		Student student = new Student(101, "Nate", subject);
		student.displayDetails();
	}
}
```

EXERCISE

```
  class Address{
private String doorNo;
private static String street;
private static String city;
private int zipcode;
Address(String doorNo, String street, String city, int zipcode){
this.doorNo= doorNo;
this.street= street;
this.city= city;
this.zipcode= zipcode;
}
public String getDoorNo() {
return doorNo;
}
public void setDoorNo(String doorNo) {
this.doorNo = doorNo;
}
public static String getStreet() {
return street;
}
public void setStreet(String street) {
this.street = street;
```

```
}
public static String getCity() {
return city;
}
public void setCity(String city) {
this.city = city;
}
public int getZipcode() {
return zipcode;
}
public void setZipcode(int zipcode) {
this.zipcode = zipcode;
}
}

class Customer{
private String customerId;
private String customerName;
private long contactNumber;
private Address address;
Customer(){
}
Customer(String customerId, String customerName, long contactNumber, Address address){
this.customerId= customerId;
this.customerName= customerName;
this.contactNumber= contactNumber;
this.address= address;
}
Customer(String customerName, long contactNumber, Address address){
this.customerName= customerName;
this.contactNumber= contactNumber;
this.address= address;
}
public String getCustomerId() {
return customerId;
}
public void setCustomerId(String customerId) {
this.customerId = customerId;
}
public String getCustomerName() {
return customerName;
}
public void setCustomerName(String customerName) {
this.customerName = customerName;
}
public long getContactNumber() {
return contactNumber;
}
public void setContactNumber(long contactNumber) {
this.contactNumber = contactNumber;
}
public Address getAddress() {
return address;
}
public void setAddress(Address address) {
this.address = address;
}
public void displayCustomerDetails(){
System.out.println("Customer Id is: " + getCustomerId());
System.out.println("Customer Name is: "+ getCustomerName());
System.out.println("Contact Number is: " + getContactNumber());
System.out.println("Address is: " + Address.getCity()+ " " + Address.getStreet());
}
public double payBill(double totalPrice)
{
totalPrice= totalPrice + 5.0;
return totalPrice;
}
  }

class Tester {

public static void main(String[] args) {
Address add= new Address("H.No. 761", "Janipur colony", "Jammu", 180007);
Customer cust= new Customer("ECE10010", "Sonakshi", 999463590L, add);
cust.displayCustomerDetails();
double x= cust.payBill(50);
System.out.println(x);


}

}
```

ASSOCIATION EXERCISE

```
class CabServiceProvider{
    private static final String String = null;
        private String cabServiceName;
        private int totalCabs;
        public CabServiceProvider(String cabServiceName, int totalCabs){
                this.cabServiceName= cabServiceName;
                this.totalCabs= totalCabs;
        }
        public String getCabServiceName() {
                return cabServiceName;
        }
        public void setCabServiceName(String cabServiceName) {
                this.cabServiceName = cabServiceName;
        }
        public int getTotalCabs() {
                return totalCabs;
        }
        public void setTotalCabs(int totalCabs) {
                this.totalCabs = totalCabs;
        }
        public double calculateRewardPrice(Driver driver){
                double bonus=0.0;
                double rewardAmount=0.0;
                if(cabServiceName.equals("Halo")){
                        if(driver.getAverageRating()>= 4.5 && driver.getAverageRating()<=5){
                                rewardAmount= 10*driver.getAverageRating();
                        }
                        else if(driver.getAverageRating()>= 4 && driver.getAverageRating()<4.5){
                                rewardAmount= 5*driver.getAverageRating();
                        }
                        else {
                                rewardAmount= 0.0;
                        }
        }
        else if(cabServiceName.equals("Aber")){
                if(driver.getAverageRating()>= 4.5 && driver.getAverageRating()<=5){
                    rewardAmount= 8*driver.getAverageRating();
                }
                else if(driver.getAverageRating()>= 4 && driver.getAverageRating()<4.5){
                    rewardAmount= 3*driver.getAverageRating();
                }
                else {
                    rewardAmount= 0.0;
                }

        }
        else{
            rewardAmount=0.0;
        }
        bonus= Math.round(rewardAmount*100.0)/100.0;
        return bonus;
    }
}

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
	}
}
```

ASSIGNMENTS

STATIC

```
1. package aggregation;
class Food {
public String foodName;
public String cuisine;
public String foodType;
public int quantityAvailable;
public double unitPrice;
}
class Order{
private static int orderCounterId;
private int orderId;
private Food[] orderedFoods;
private double totalPrice;
private String status;
static{
orderCounterId= 100;
}
public Order(){
}
public Order(Food[] orderedFoods){
this.orderedFoods= orderedFoods;
orderId= ++orderCounterId;
}
public static int getOrderCounterId() {
return orderCounterId;
}
public static void setOrderCounterId(int orderCounterId) {
Order.orderCounterId = orderCounterId;
}
public int getOrderId() {
return orderId;
}
public void setOrderId(int orderId) {
this.orderId = orderId;
}
public Food[] getOrderedFoods() {
```

```
return orderedFoods;
}
public void setOrderedFoods(Food[] orderedFoods) {
this.orderedFoods = orderedFoods;
}
public double getTotalPrice() {
return totalPrice;
}
public void setTotalPrice(double totalPrice) {
this.totalPrice = totalPrice;
}
public String getStatus() {
return status;
}
public void setStatus(String status) {
this.status = status;
}
public static int getTotalNoOfOrders(){
return(Order.orderCounterId - 100);
}
public double calculateTotalPrice(String paymentMode) {
double foodPrice = 0;
double finalPrice = 0;
float serviceCharge = 0f;
int unitPrice = 100;
foodPrice+=unitPrice*1;
if (paymentMode.equals("Credit Card") || paymentMode.equals("Debit Card")) {
serviceCharge = 2.0f;
}
else if (paymentMode.equals("PayPal")) {
serviceCharge = 2.9f;
}
finalPrice = foodPrice+foodPrice*(serviceCharge/100);
this.setTotalPrice(finalPrice);
return finalPrice;
}
}
class Tester2 {
public static void main(String args[]) {
Food food1 = new Food();
Food food2 = new Food();
Food food3 = new Food();
Food food4 = new Food();
Food food5 = new Food();

Food[] totalFoods = { food1, food2, food3, food4, food5 };

Order order= new Order();
System.out.println(order.calculateTotalPrice("Credit Card"));
System.out.println(order.getOrderedFoods());
}
}
```

2.

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

```
        public Participant(String name, long contactNumber, String city){
                this.name= name;
                this.contactNumber= contactNumber;
                this.city= city;
                registrationId= "D" + ++counter;
        }
        static{
                counter=10000;
        }
        public static int getCounter() {
                return counter;
        }
        public static void setCounter(int counter) {
                Participant.counter = counter;
        }
        public String getRegistrationId() {
                return registrationId;
        }
        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }
        public String getCity() {
                return city;
        }
        public void setCity(String city) {
                this.city = city;
        }
        public long getContactNumber() {
                return contactNumber;
        }
        public void setContactNumber(long contactNumber) {
                this.contactNumber = contactNumber;
        }

}
  class AssignmentStatic2 {

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());
                }

        }

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

}

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());
                        }
                  }
        }
}
-----------------------
-----------------------------
```

ASSIGNMENTS

AGGREGATION

```
1. class Food {
private String foodName;
private String cuisine;
private String foodType;
private int quantityAvailable;
private double unitPrice;
public String getFoodName() {
return foodName;
}
public void setFoodName(String foodName) {
this.foodName = foodName;
}
public String getCuisine() {
return cuisine;
}
public void setCuisine(String cuisine) {
this.cuisine = cuisine;
}
public String getFoodType() {
return foodType;
}
public void setFoodType(String foodType) {
this.foodType = foodType;
}
public int getQuantityAvailable() {
return quantityAvailable;
}
public void setQuantityAvailable(int quantityAvailable) {
this.quantityAvailable = quantityAvailable;
}
public double getUnitPrice() {
return unitPrice;
}
public void setUnitPrice(double unitPrice) {
this.unitPrice = unitPrice;
}
}

class Address{
private String doorNo;
private static String street;
private static String city;
private int zipcode;
Address(String doorNo, String street, String city, int zipcode){
this.doorNo= doorNo;
Address.street= street;
Address.city= city;
this.zipcode= zipcode;
```

```
}
public String getDoorNo() {
return doorNo;
}
public void setDoorNo(String doorNo) {
this.doorNo = doorNo;
}
public static String getStreet() {
return street;
}
public void setStreet(String street) {
Address.street = street;
}
public static String getCity() {
return city;
}
public void setCity(String city) {
Address.city = city;
}
public int getZipcode() {
return zipcode;
}
public void setZipcode(int zipcode) {
this.zipcode = zipcode;
}
}

class Customer{
private String customerId;
private String customerName;
private long contactNumber;
private Address address;
public Customer(){
}
public Customer(String customerId,String customerName,long contactNumber,Address
address){
this.customerId= customerId;
this.customerName= customerName;
this.contactNumber= contactNumber;
this.address= address;
}
public Customer(String customerName,long contactNumber,Address address){
this.customerName= customerName;
this.contactNumber= contactNumber;
this.address= address;
}
public String getCustomerId() {
return customerId;
}
public void setCustomerId(String customerId) {
this.customerId = customerId;
}
public String getCustomerName() {
return customerName;
}
public void setCustomerName(String customerName) {
this.customerName = customerName;
}
public long getContactNumber() {
return contactNumber;
```

```
}
public void setContactNumber(long contactNumber) {
this.contactNumber = contactNumber;
}
public Address getAddress() {
return address;
}
public void setAddress(Address address) {
this.address = address;
}
public boolean generateBill(Order order) {
System.out.println("Bill details \n***********");
System.out.println("Items ordered : ");
for (Food food : order.getOrderedFoods()) {
System.out.println(food.getFoodName());
}
double payableAmount = order.calculateTotalPrice("Credit Card");
System.out.println("Payable Amount : $" + (int) (payableAmount * 100)
/ 100.0);
return true;
}
public void displayCustomerDetails(){
System.out.println("Customer Id is: " + getCustomerId());
System.out.println("Customer Name is: "+ getCustomerName());
System.out.println("Contact Number is: " + getContactNumber());
System.out.println("Address is: " + Address.getCity()+ " " + Address.getStreet());
}
}
class Order{
private static int orderCounterId;
private int orderId;
private Food[] orderedFoods;
private double totalPrice;
private String status;
private Customer customer;
public Customer getCustomer() {
return customer;
}
public void setCustomer(Customer customer) {
this.customer = customer;
}
static{
orderCounterId= 100;
}
public Order(){
}
public Order(Food[] orderedFoods, Customer customer){
this.orderedFoods= orderedFoods;
this.customer= customer;
orderId= ++orderCounterId;
}
public static int getOrderCounterId() {
return orderCounterId;
}
public static void setOrderCounterId(int orderCounterId) {
Order.orderCounterId = orderCounterId;
}
public int getOrderId() {
return orderId;
}
```

```
public void setOrderId(int orderId) {
this.orderId = orderId;
}
public Food[] getOrderedFoods() {
return orderedFoods;
}
public void setOrderedFoods(Food[] orderedFoods) {
this.orderedFoods = orderedFoods;
}
public double getTotalPrice() {
return totalPrice;
}
public void setTotalPrice(double totalPrice) {
this.totalPrice = totalPrice;
}
public String getStatus() {
return status;
}
public void setStatus(String status) {
this.status = status;
}
public static int getTotalNoOfOrders(){
return(Order.orderCounterId - 100);
}
public double calculateTotalPrice(String paymentMode) {
double foodPrice = 0;
double finalPrice = 0;
float serviceCharge = 0f;
for (Food food : orderedFoods) {
foodPrice+=food.getUnitPrice()*1;
}
if (paymentMode.equals("Credit Card") || paymentMode.equals("Debit Card")) {
serviceCharge = 2.0f;
}
else if (paymentMode.equals("PayPal")) {
serviceCharge = 2.9f;
}
finalPrice = foodPrice+foodPrice*(serviceCharge/100);
this.setTotalPrice(finalPrice);
return finalPrice;
}
}
public class Tester {

}
```

```
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 double price;
private int quantity;
private Author author;
public Book(String name, double price, int quantity, Author author){
this.name=name;
this.price=price;
this.quantity=quantity;
this.author=author;
}
public void displayAuthorDetails(){
System.out.println("Displaying author details");
System.out.println("Author Name: "+ author.getName());
System.out.println("Author Email Id: "+ author.getEmailId());
System.out.println("Author Gender: "+ author.getGender());
}
public String getName() {
return name;
}
public void setName(String name) {
this.name = name;
}
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 Author getAuthor() {
return author;
}
public void setAuthor(Author author) {
this.author = author;
}
}
class Tester1 {
public static void main(String args[]){
Author author1= new Author("Joshua Bloch", "joshua@email.com", 'M');
```

```
Book book1= new Book("Effective Java 1", 45.0, 15, author1);
book1.displayAuthorDetails();
}
}
----------------------
3. class Room {

	//Implement your code here

	//Uncomment the below method after implementation before verifying
	//DO NOT MODIFY THE METHOD
	private int roomNo;
	private int capacity;
	private static int roomCounter;
	static{
		roomCounter= 499;
	}
	public Room(){
		capacity=4;
		roomNo= ++roomCounter;
	}

	public int getCapacity() {
		return 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;
	}


	public String toString(){
		return "Room\nroomNo: "+this.roomNo+"\ncapacity: "+this.capacity;
	}


}

class Member {
//Implement your code here

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


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


}

class Admin {
//Implement your code here
    public void assignRoom(Room[] rooms, Member member){
        for(Room room: rooms){
            if(room.getCapacity()>0 && room.getCapacity()<=4){
                member.setRoom(room);
                room.setCapacity(room.getCapacity()-1);
                break;
            }
        }
    }
}


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");
            }
        }
    }
}
```


DAY 2

CONSTRUCTOR CALL IN INHERITANCE- TRYOUT

```
class Employee {
    Employee() {
        System.out.println("Employee constructor invoked");
    }
}

class Manager extends Employee {
    Manager() {
        System.out.println("Manager constructor invoked");
    }
}

class Tester {
    public static void main(String[] args) {
        Manager manager = new Manager();
    }
}
```

NEED FOR SUPER CONSTRUCTORS= TRYOUT

```
class Customer {
    }

    private String customerId;
    private String customerName;

    public Customer(String customerId, String customerName) {
        this.customerId = customerId;
        this.customerName = customerName;
    }

    public Customer() {
        System.out.println("Parent parameterless constructor");
    }

    public String getCustomerId() {
        return customerId;
    }

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

    public String getCustomerName() {
        return customerName;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public void displayCustomerDetails() {
        System.out.println("Displaying customer details \n**************************");
        System.out.println("Customer Id : " + this.customerId);
        System.out.println("Customer Name : " + this.customerName);
        System.out.println();
    }

}

class RegularCustomer extends Customer {

    private float discount;

    public RegularCustomer(String custId, String custName) {

        this.setCustomerId(custId);
        this.setCustomerName(custName);
        this.discount = 5.0f;
        System.out.println("Child parameterized constructor");
    }

    public float getDiscount() {
        return discount;
    }

    public void setDiscount(float discount) {
        this.discount = discount;
    }

}
```

```
class Tester {

	public static void main(String[] args) {

		RegularCustomer regularCustomer = new RegularCustomer("C1010", "Johns Kora");
		regularCustomer.displayCustomerDetails();

	}
}
```

INHERITANCE TRYOUT

```
class Employee {
	int employeeId;
	String employeeName;

	// Parameterized constructor
	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 void display() {
		System.out.println("Employee details");
		System.out.println("Employee Id: " + employeeId);
		System.out.println("Employee Name: " + employeeName);
	}
}

class Manager extends Employee {
	private String designation;

	Manager(int employeeId, String employeeName, String designation) {
		super(employeeId, employeeName);
		this.designation = designation;
	}

	public String getDesignation() {
		return designation;
	}

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

class Tester {
	public static void main(String[] args) {
		Manager obj = new Manager(101, "John", "Lead");
		obj.display();
		System.out.println("Designation: " + obj.getDesignation());
	}
}
--------------------
```

INHERITANCE-EXERCISE 1

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

	public Camera() {
		this.brand = "Nikon";
	}
	public Camera(String brand, double cost){
		this.brand=brand;
		this.cost= cost;
	}

	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) {
		super(brand, cost);
		this.memory = 16;
	}

	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- TRYOUT*
------
class Shape {

	// Method to find the area of circle
	public float calculateArea(float radius) {
		return 3.14f * radius * radius;
	}

	// Method to find the area of rectangle
	public float calculateArea(float length, float breadth) {
		return length * breadth;
	}

	// Code another overloaded method to find the area of triangle
}

class Tester {

	public static void main(String[] args) {

		Shape shape = new Shape();

		float circleArea = shape.calculateArea(1.7f);
		float rectangleArea = shape.calculateArea(2.5f, 3.4f);

		System.out.println("Area of circle: " + circleArea);
		System.out.println("Area of rectangle: " + rectangleArea);

		// Invoke the method to find the area of triangle
		// Display the area of triangle
	}
}
-----------------------
*METHOD OVERLOADING-EXERCISE 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 x2= this.xCoordinate, x1=0.0, y2= this.yCoordinate, y1=0.0;
		double d= ((x2-x1)*(x2-x1)+ (y2-y1)*(y2-y1));
		double distance= Math.sqrt(d);
		return Math.round(distance*100.0)/100.0;
	}
	public double calculateDistance(Point point){
		double    x2=    this.xCoordinate,    x1=point.xCoordinate,    y2=    this.yCoordinate,
y1=point.yCoordinate;
		double d= ((x2-x1)*(x2-x1)+ (y2-y1)*(y2-y1));
		double distance= Math.sqrt(d);
		return Math.round(distance*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

	}
}
```

INHERITANCE-ASSIGNEMENT 1(INCORRECT)

```
class Employee {

   //Implement your code here

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

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


class PermanentEmployee extends Employee {

    //Implement your code here

        //Uncomment the below method after implementation before verifying
        //DO NOT MODIFY THE METHOD
        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 void calculateMonthlySalary() {
                float experienceComponentPercentage = 0f;

                if (this.experience >= 3 && this.experience < 5) {
                        experienceComponentPercentage = 5;
                } else if (this.experience >= 5 && this.experience < 10) {
                        experienceComponentPercentage = 7;
                } else if (this.experience >= 10) {
                        experienceComponentPercentage = 12;
                }

                double salary = (float) ((this.basicPay * (experienceComponentPercentage/100)) + this.basicPay + this.hra);
                this.setSalary(salary);
        }

        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 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 {

		//Implement your code here

	//Uncomment the below method after implementation before verifying
	//DO NOT MODIFY THE METHOD
	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 void calculateSalary(){
		setSalary(hoursWorked*wage);

	}

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 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 ASSIGNMENT 1

```
class Bill{
        //Implement your code here
        public double findPrice(int itemId){
                double price=0;
                switch (itemId){
                        case 1001: price=25;
                        break;
                          case 1002: price=20;
                        break;
                          case 1003: price=23;
                        break;
                          case 1004: price=18;
                        break;
                        default: price=0;
                }
                return price;
        }
        public double findPrice(String brandName, String itemType, int size){
                double price=0;
                if(brandName.equals("Puma")){
                        if(itemType.equals("T-shirt")){
                                if(size==34 || size==36){
                                        price=25;
                                }
                        }
                        else if(itemType.equals("Skirt")){
                                  if(size==38 || size==40){
                                        price=20;
                                }
                        }
                }
                else if(brandName.equals("Reebok")){
                        if(itemType.equals("T-shirt")){
                                if(size==34 || size==36){
                                        price=23;
                                }
```

```
                    }
                    else if(itemType.equals("Skirt")){
                         if(size==38 || size==40){
                              price=18;
                         }
                    }
               }
               return price;
          }
     }

     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 ASSIGNMENT 2

```
class Point{
    //Reuse the code of Method Overloading - Exercise 1

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

     private double xCoordinate;
     private double yCoordinate;
     public Point(double xCoordinate, double yCoordinate){
          this.xCoordinate= xCoordinate;
          this.yCoordinate= yCoordinate;
     }
     public double calculateDistance(){
          double x2= this.xCoordinate, x1=0.0, y2= this.yCoordinate, y1=0.0;
          double d= ((x2-x1)*(x2-x1)+ (y2-y1)*(y2-y1));
          double distance= Math.sqrt(d);
          return Math.round(distance*100.0)/100.0;
     }
     public double calculateDistance(Point point){
          double   x2=   this.xCoordinate,   x1=point.xCoordinate,   y2=   this.yCoordinate,
y1=point.yCoordinate;
          double d= ((x2-x1)*(x2-x1)+ (y2-y1)*(y2-y1));
          double distance= Math.sqrt(d);
          return Math.round(distance*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;
	}

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

class Triangle {
      //Implement your code here
	private Point point1;
	private Point point2;
	private Point point3;
public Triangle(){
	point1= new Point(0,0);
	point2= new Point(1,1);
	point3= new Point(2,5);
}
public		Triangle(double		point1XCoordinate,double		point1YCoordinate,double
point2XCoordinate,double		point2YCoordinate,double		point3XCoordinate,double
point3YCoordinate){
	point1= new Point(point1XCoordinate,point1YCoordinate);
	point2= new Point(point2XCoordinate,point2YCoordinate);
	point3= new Point(point3XCoordinate,point3YCoordinate);
}
public Triangle(Point point1, Point point2, Point point3){
	this.point1= point1;
	this.point2= point2;
	this.point3= point3;
}

public double calculatePerimeter(){
	double perimeter,a,b,c;
	a= point1.calculateDistance(point2);
	b= point2.calculateDistance(point3);
	c= point3.calculateDistance(point1);
	perimeter= a+b+c;
	return Math.round(perimeter*100)/100.0;
}

public double calculateArea(){
	double area,s,a,b,c;
	a= point1.calculateDistance(point2);
	b= point2.calculateDistance(point3);
	c= point3.calculateDistance(point1);
	s=(a+b+c)/2;
	area= Math.sqrt(s*(s-a)*(s-b)*(s-c));
	return Math.round(area*100)/100.0;
}
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;
}
}



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

	}
}
```


DAY 3

METHOD OVERRIDING- TRYOUT
----
1. class Tester {

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

		 Customer customer = new Customer();
		 System.out.println("Final bill amount: "+customer.payBill(40.0));
```

```
        // Parent Reference -> Parent Object

         RegularCustomer regularCustomer = new RegularCustomer();
         System.out.println("Final bill amount: "+regularCustomer.payBill(40.0));
        // Child Reference -> Child Object

        Customer regCust = new RegularCustomer();
        // Parent Reference -> Child Object
        System.out.println("Final Bill : " + regCust.payBill(40.0));
    }
}

class Customer {

    public double payBill(double totalPrice) {
        System.out.println("Final bill for the customer is calculated here");
        return totalPrice;
    }
}

class RegularCustomer extends Customer {

    @Override
    public double payBill(double totalPrice) {
        System.out.println("Final bill for the regular customer is calculated here");
        double priceAfterDiscount = totalPrice * (1 - (5f / 100));
        return priceAfterDiscount;
    }

}
```

```
2. class Tester {
    public static void main(String args[]) {
        Bank bank;
        bank = new ABCBank();
        System.out.println("ABCBank - Rate of Interest(%): "
                + bank.getRateOfInterest());
        bank = new DEFBank();
        System.out.println("DEFBank - Rate of Interest(%): "
                + bank.getRateOfInterest());
        bank = new GHIBank();
        System.out.println("GHIBank - Rate of Interest(%): "
                + bank.getRateOfInterest());
    }
}

class Bank {
    public float getRateOfInterest() {
        return 0;
    }
    // remove the access specifier and observe the output
}

class ABCBank extends Bank {
    public float getRateOfInterest() {
        return 8.99f;
    }
    // Keeping the parent access specifier as public, remove the child access
    // specifier and observe the output
}

class DEFBank extends Bank {
	public float getRateOfInterest() {
		return 9.4f;
	}
	// Keeping the parent access specifier as public, change the child access
	// specifier to private/protected and observe the output
}

class GHIBank extends Bank {
	public float getRateOfInterest() {
		return 8.1f;
	}
}
```

GENERIC METHOD TRYOUT

```
class DynamicBindingTester {
	public static void main(String[] args) {
		Employee employee = new Employee();
		Manager manager = new Manager();
		Employee eduEmployee = new Educator();
		Educator managerEdu = new Manager();

		displayEmployeeDetails(employee);
		displayEmployeeDetails(manager);
		displayEmployeeDetails(eduEmployee);
		displayEmployeeDetails(managerEdu);
	}

	// Employee reference can accept its object and any of the child object
	public static void displayEmployeeDetails(Employee employee) {
		employee.displayDetails(); // displayDetails invoked will be based on the object
received
	}
}

class Employee {
	String name = "James Anthony";

	public void displayDetails(){
		System.out.println(name+" is an employee");
	}
}
class Educator extends Employee {
	public void displayDetails(){
		System.out.println(name+" is an educator");
	}
}
class Manager extends Educator {
	public void displayDetails(){
		System.out.println(name+" is a manager");
	}
}
```

METHOD OVERRIDING EXERCISE

```
class User{
	//Implement your code here
	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 boolean makePayment(double billAmount){
		if(this.getWalletBalance()>=billAmount){
			double x= (double)(this.getWalletBalance()-billAmount);
			this.setWalletBalance(x);
			return true;
		}
		else{
			return false;
		}
	}
	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;
	}

}

class PremiumUser extends User{
	// Implement your code here
	private int rewardPoints;
	public int getRewardPoints() {
		return rewardPoints;
	}
	public void setRewardPoints(int rewardPoints) {
		this.rewardPoints = rewardPoints;
	}
	public PremiumUser(int id,String userName, String emailId,double walletBalance){
		super(id, userName, emailId, walletBalance);
	}
	@Override
public boolean makePayment(double billAmount){
        if(this.getWalletBalance()>=billAmount){
            int y= (int)(this.rewardPoints+((billAmount*10)/100));
            this.setRewardPoints(y);
            double z= (double)(this.getWalletBalance()-billAmount);
            this.setWalletBalance(z);
            return true;
        }
        else{
            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();
        }
}
```

EQUAL OBJECTS TRYOUT

```
class Food {

        private String foodName;
        private String cuisine;
        private String foodType;
```

```
	private int quantityAvailable;
	private double unitPrice;

	public String getFoodName() {
		return foodName;
	}

	public void setFoodName(String foodName) {
		this.foodName = foodName;
	}

	public String getCuisine() {
		return cuisine;
	}

	public void setCuisine(String cuisine) {
		this.cuisine = cuisine;
	}

	public String getFoodType() {
		return foodType;
	}

	public void setFoodType(String foodType) {
		this.foodType = foodType;
	}

	public int getQuantityAvailable() {
		return quantityAvailable;
	}

	public void setQuantityAvailable(int quantityAvailable) {
		this.quantityAvailable = quantityAvailable;
	}

	public double getUnitPrice() {
		return unitPrice;
	}

	public void setUnitPrice(double unitPrice) {
		this.unitPrice = unitPrice;
	}
}

class Tester {

	public static void main(String[] args) {
		Food foodOne = new Food();
		foodOne.setFoodName("Sandwich");
		foodOne.setCuisine("Continental");
		foodOne.setFoodType("Veg");
		foodOne.setQuantityAvailable(100);
		foodOne.setUnitPrice(10);

		Food foodTwo = new Food();
		foodTwo.setFoodName("Sandwich");
		foodTwo.setCuisine("Continental");
		foodTwo.setFoodType("Veg");
		foodTwo.setQuantityAvailable(200);
		foodTwo.setUnitPrice(10);
```

```
        if (foodOne == foodTwo) {
            System.out.println("The food objects are same!");
        } else {
            System.out.println("The food objects are different!");
        }
    }
}
---------------
*EQUALS TRYOUT*
class Food {

    private String foodName;
    private String cuisine;
    private String foodType;
    private int quantityAvailable;
    private double unitPrice;

    public String getFoodName() {
        return foodName;
    }

    public void setFoodName(String foodName) {
        this.foodName = foodName;
    }

    public String getCuisine() {
        return cuisine;
    }

    public void setCuisine(String cuisine) {
        this.cuisine = cuisine;
    }

    public String getFoodType() {
        return foodType;
    }

    public void setFoodType(String foodType) {
        this.foodType = foodType;
    }

    public int getQuantityAvailable() {
        return quantityAvailable;
    }

    public void setQuantityAvailable(int quantityAvailable) {
        this.quantityAvailable = quantityAvailable;
    }

    public double getUnitPrice() {
        return unitPrice;
    }

    public void setUnitPrice(double unitPrice) {
        this.unitPrice = unitPrice;
    }

    // equals method of Object class overridden for comparing two Food objects
    // based on foodName and foodType
    @Override
    public boolean equals(Object obj) {
        Food otherFood = (Food) obj;
        if (this.foodName.equals(otherFood.foodName)) {
            if (this.foodType.equals(otherFood.foodType))
                return true;
        }
        return false;
    }
}

class Tester {

    public static void main(String[] args) {
        Food foodOne = new Food();
        foodOne.setFoodName("Sandwich");
        foodOne.setCuisine("Continental");
        foodOne.setFoodType("Veg");
        foodOne.setQuantityAvailable(100);
        foodOne.setUnitPrice(10);

        Food foodTwo = new Food();
        foodTwo.setFoodName("Sandwich");
        foodTwo.setCuisine("Continental");
        foodTwo.setFoodType("Veg");
        foodTwo.setQuantityAvailable(200);
        foodTwo.setUnitPrice(10);

        if (foodOne.equals(foodTwo)) {
            System.out.println("foodOne and foodTwo are same!");
        } else {
            System.out.println("foodOne and foodTwo are different!");
        }

        Food foodThree = new Food();
        foodThree.setFoodName("Burger");
        foodThree.setCuisine("Continental");
        foodThree.setFoodType("Veg");
        foodThree.setQuantityAvailable(100);
        foodThree.setUnitPrice(10);

        if (foodOne.equals(foodThree)) {
            System.out.println("foodOne and foodThree are same!");
        } else {
            System.out.println("foodOne and foodThree are different!");
        }
    }
}
```

HASHCODE TRYOUT

```
class Food {

    private String foodName;
    private String cuisine;
    private String foodType;
    private int quantityAvailable;
    private double unitPrice;

    public String getFoodName() {
        return foodName;
```

```
	}

	public void setFoodName(String foodName) {
		this.foodName = foodName;
	}

	public String getCuisine() {
		return cuisine;
	}

	public void setCuisine(String cuisine) {
		this.cuisine = cuisine;
	}

	public String getFoodType() {
		return foodType;
	}

	public void setFoodType(String foodType) {
		this.foodType = foodType;
	}

	public int getQuantityAvailable() {
		return quantityAvailable;
	}

	public void setQuantityAvailable(int quantityAvailable) {
		this.quantityAvailable = quantityAvailable;
	}

	public double getUnitPrice() {
		return unitPrice;
	}

	public void setUnitPrice(double unitPrice) {
		this.unitPrice = unitPrice;
	}

	// equals method of Object class overridden for comparing two Food objects
	// based on foodName and foodType
	@Override
	public boolean equals(Object obj) {
		Food otherFood = (Food) obj;
		if (this.foodName.equals(otherFood.foodName)) {
			if (this.foodType.equals(otherFood.foodType))
				return true;
		}
		return false;
	}

	// hashCode method overridden
	@Override
	public int hashCode() {
		int result = 1;
		result = result + (foodName.hashCode());
		return result;
	}
}

class Tester {
	public static void main(String[] args) {
		Food foodOne = new Food();
		foodOne.setFoodName("Sandwich");
		foodOne.setCuisine("Continental");
		foodOne.setFoodType("Veg");
		foodOne.setQuantityAvailable(100);
		foodOne.setUnitPrice(10);

		Food foodTwo = new Food();
		foodTwo.setFoodName("Sandwich");
		foodTwo.setCuisine("Continental");
		foodTwo.setFoodType("Veg");
		foodTwo.setQuantityAvailable(200);
		foodTwo.setUnitPrice(10);

		if (foodOne.equals(foodTwo)) {
			System.out.println("foodOne and foodTwo are same!");
		} else {
			System.out.println("foodOne and foodTwo are different!");
		}

		System.out.println("Hash code for foodOne : " + foodOne.hashCode());
		System.out.println("Hash code for foodTwo : " + foodTwo.hashCode());

		Food foodThree = new Food();
		foodThree.setFoodName("Burger");
		foodThree.setCuisine("Continental");
		foodThree.setFoodType("Veg");
		foodThree.setQuantityAvailable(100);
		foodThree.setUnitPrice(10);

		if (foodOne.equals(foodThree)) {
			System.out.println("foodOne and foodThree are same!");
		} else {
			System.out.println("foodOne and foodThree are different!");
		}

		System.out.println("Hash code for foodOne : " + foodOne.hashCode());
		System.out.println("Hash code for foodThree : " + foodThree.hashCode());

	}
}
```
----

to-STRING TRYOUT

```
class Food {

	private String foodName;
	private String cuisine;
	private String foodType;
	private int quantityAvailable;
	private double unitPrice;

	public String getFoodName() {
		return foodName;
	}

	public void setFoodName(String foodName) {
		this.foodName = foodName;
	}
```

```
public String getCuisine() {
      return cuisine;
}

public void setCuisine(String cuisine) {
      this.cuisine = cuisine;
}

public String getFoodType() {
      return foodType;
}

public void setFoodType(String foodType) {
      this.foodType = foodType;
}

public int getQuantityAvailable() {
      return quantityAvailable;
}

public void setQuantityAvailable(int quantityAvailable) {
      this.quantityAvailable = quantityAvailable;
}

public double getUnitPrice() {
      return unitPrice;
}

public void setUnitPrice(double unitPrice) {
      this.unitPrice = unitPrice;
}

// equals method of Object class overridden for comparing two Food objects
// based on foodName and foodType
@Override
public boolean equals(Object obj) {
      Food otherFood = (Food) obj;
      if (this.foodName.equals(otherFood.foodName)) {
            if (this.foodType.equals(otherFood.foodType))
                  return true;
      }
      return false;
}

// hashCode method overridden
@Override
public int hashCode() {
      int result = 1;
      result = result + (foodName.hashCode());
      return result;
}

// toString is overridden to provide a custom textual representation
@Override
public String toString() {
      return "Food -> " + "Food name: " + this.foodName + ", Cuisine: "
                  + this.cuisine + ", Food type: " + this.foodType
                  + ", Quantity avaialable: " + this.quantityAvailable
                  + ", unitPrice: " + unitPrice;
```

```
        }
}

class Tester {

        public static void main(String[] args) {
                Food foodOne = new Food();
                foodOne.setFoodName("Sandwich");
                foodOne.setCuisine("Continental");
                foodOne.setFoodType("Veg");
                foodOne.setQuantityAvailable(100);
                foodOne.setUnitPrice(10);

                // Comment the toString() in the Food class and execute the code
                System.out.println(foodOne);
        }
}
```


WRAPPER CLASSES TRYOUT

```
class Tester {

        public static void main(String[] args) {

                // Comparison of Integer objects
                Integer x = 5;
                Integer y = new Integer(5); // int data converted to Integer object
                System.out.println("x == y: " + (x == y));
                System.out.println("x.equals(y): " + x.equals(y));

                // Wrap the primitive content into wrapper class objects
                Integer number = Integer.valueOf(x);
                System.out.println("Wrapping x to its Integer: " + number);
                System.out.println("Check if number is of type Integer: "
                                + (number instanceof Integer));

                // Wrappers can be used to convert numeric strings to numeric datatypes
                String numStr = "123";
                int numInt = Integer.parseInt(numStr);
                System.out.println("String to integer: " + numInt);

                String doubleStr = "123.45";
                double numDouble = Double.parseDouble(doubleStr);
                System.out.println("String to double: " + numDouble);

                // Type casting cannot be used to convert any wrapper type to another,
                // it will give compilation error
                /* Uncomment the below code and observe the output*/
                // Long phoneNoLong = (Long) new Integer(44281234);

                // We can make use of methods like intValue(), byteValue(),
                // floatValue(), etc. for conversion
                Integer phoneNo = 44281234;
                // longValue() converts the Integer value to long data type
                Long phoneNoLong = phoneNo.longValue();
                System.out.println("Integer to Long: " + phoneNoLong);

                // Converts the Integer object to binary value
                System.out.println("Integer 5 as binary string: "
                        + Integer.toBinaryString(5));
      }
}
```

BLANK FINAL VARIABLE TRYOUT
```
class Demo {
      final int num; // blank final variable

      public Demo() {
            num = 10;
      }

      public void displayNumber() {
            System.out.println(num);
      }
}

class Tester {
      public static void main(String args[]) {
            Demo demo = new Demo();
            demo.displayNumber();
      }
}
```

FINAL EXERCISE 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;
            }
            int good= (int)(bonus+ this.STIPEND);
            return good;
      }
}
```

```
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);
	}

}
```


DAY 4

ABSTRACT CLASS AND METHODS-TRYOUT
```
abstract class GrandParent {
	abstract void display();
}

abstract class Parent extends GrandParent { // Can we remove the abstract keyword from
here?
	final void displayInParent() {
		System.out.println("In Parent");
	}
}

class Child extends Parent {
	void display() {
		System.out.println("Child completes Parent and GrandParent");
		super.displayInParent();
	}
}

final class GrandChild extends Child {
	void display() {
		System.out.println("In GrandChild");
		super.display();
	}
}

//Uncomment the code given below and observe
//class GreatGrandChild extends GrandChild { }

class Tester {
	public static void main(String[] args) {
		new GrandChild().display();
	}
}
```

INSTANCE OF- TRYOUT

```
abstract class Employee{
	private String employeeId;
	private String name;
	private static int counter;

	static{
		counter=101;
	}

	public Employee(String name){
		//Checking the type of the current instance
		if(this instanceof PermanentEmployee)
			employeeId="P"+counter++;
		else if(this instanceof ContractEmployee)
			employeeId="C"+counter++;
		setName(name);
	}

	public abstract void calculateSalary();

	public String getEmployeeId(){
		return employeeId;
	}

	public String getName(){
		return name;
	}

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

class PermanentEmployee extends Employee{

	public PermanentEmployee(String name){
		super(name);
	}

	@Override
	public void calculateSalary(){
		System.out.println("Calculating salary of PermanentEmployee");
	}

	public void calculateBonus(){
		System.out.println("Calculating bonus of PermanentEmployee");
	}
}


class ContractEmployee extends Employee{

	public ContractEmployee(String name){
		super(name);
	}

	@Override
	public void calculateSalary(){
        System.out.println("Calculating salary of ContractEmployee");
    }

}

class SalarySlipGenerator{
    public void displaySalarySlip(Employee employee){
        employee.calculateSalary();
        //Checking if employee is an instance of PermanentEmployee
        if(employee instanceof PermanentEmployee){
            //Type casting parent class reference to child class for accessing child class
method
            PermanentEmployee permanentEmployee=(PermanentEmployee)employee;
            permanentEmployee.calculateBonus();
        }
    }
}


class Tester{
    public static void main(String[] args) {
        PermanentEmployee permanentEmployee=new PermanentEmployee("Angie");
        System.out.println("Details of permanent employee");
        System.out.println("Employee Id: "+permanentEmployee.getEmployeeId());
        System.out.println("Name: "+permanentEmployee.getName());

        System.out.println();

        ContractEmployee contractEmployee=new ContractEmployee("Roger");
        System.out.println("Details of contract employee");
        System.out.println("Employee Id: "+contractEmployee.getEmployeeId());
        System.out.println("Name: "+contractEmployee.getName());

        System.out.println();

        SalarySlipGenerator salarySlipGenerator=new SalarySlipGenerator();
        System.out.println("Salary of permanent employee");
        salarySlipGenerator.displaySalarySlip(permanentEmployee);

        System.out.println();

        System.out.println("Salary of contract employee");
        salarySlipGenerator.displaySalarySlip(contractEmployee);
    }
}
```

ABSTRACT EXERCISE 1

```
abstract class Student {
    //Implement your code here
  private String studentName;
    private int [] testScores;
    private String testResult;
    public Student(String studentName) {
        this.studentName = studentName;
        testScores=new int[4];
    }
    abstract public void generateResult() ;

    public String getStudentName() {
```

```
		return studentName;
	}
	public void setStudentName(String studentName) {
		this.studentName = studentName;
	}
	public int[] getTestScores() {
		return testScores;
	}
	public void setTestScore(int testNumber,int testScore) {
		this.testScores[testNumber] = testScore;
	}
	public String getTestResult() {
		return testResult;
	}
	public void setTestResult(String testResult) {
		this.testResult = testResult;
	}

}


class UndergraduateStudent extends Student {
	public UndergraduateStudent(String studentName) {
		super(studentName);

	}

public void generateResult() {
	int []testScores=super.getTestScores();
	int total = 0;
	for(int i=0;i<testScores.length;i++) {
		total+=testScores[i];
	}
	double average=total/testScores.length;
	if(average>=60) {
		super.setTestResult("Pass");
	}else if(average<60) {
		super.setTestResult("Fail");
	}
	}
}


class GraduateStudent extends Student {

	public GraduateStudent(String studentName) {
		super(studentName);

	}
	public void generateResult() {
		int []testScores=super.getTestScores();
		int total = 0;
		for(int i=0;i<testScores.length;i++) {
			total+=testScores[i];
		}
		double average=total/testScores.length;
		if(average>=70) {
			super.setTestResult("Pass");
		}else if(average<70) {
			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
    }
}
```

INTERFACE TRYOUT
```
interface DemoOne {
    int number = 5;
}

interface DemoTwo extends DemoOne{
    void display();
}

class DemoClassOne implements DemoTwo {
    public void display() {
        System.out.println(number);
    }
}

class Tester {
    public static void main(String[] args) {
        DemoTwo obj = new DemoClassOne();
        obj.display();
    }
}
```

INTERFACE EXERCISE 1

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

class PurchaseDetails implements Tax{
	private String purchaseId;
	private String paymentType;
	private double 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 getTaxPercentage() {
		return taxPercentage;
	}
	public void setTaxPercentage(double taxPercentage) {
		this.taxPercentage = taxPercentage;
	}

	public PurchaseDetails(String purchaseId,String paymentType) {
		this.purchaseId=purchaseId;
		this.paymentType=paymentType;
	}
	public double calculateTax(double price) {
		double total=0;
		if(this.paymentType.equals("Debit Card")) {
			total=price+(price*0.02);
			this.setTaxPercentage(2);
		}
		else if(this.paymentType.equals("Credit Card")) {
			total=price+(price*0.03);
			this.setTaxPercentage(3);
		}
		else {
			total=price+(price*0.04);
			this.setTaxPercentage(4);
		}
		return total;
	}
}

class Seller implements Tax{
	private String location;
	private double taxPercentage;
	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 Seller(String location) {
		this.location=location;
	}

	public double calculateTax(double price) {
		double tax=0;
		if(this.location.equals("Middle east")) {
			tax=price*0.15;
			this.setTaxPercentage(15);
		}
		else if(this.location.equals("Europe")) {
			tax=price*0.25;
			this.setTaxPercentage(25);
		}
		else if(this.location.equals("Canada")) {
			tax=price*0.22;
			this.setTaxPercentage(22);
		}
		else if(this.location.equals("Japan")) {
			tax=price*0.12;
			this.setTaxPercentage(12);
		}
		return tax;
	}
}
  class InterfaceExercise {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		System.out.println("Purchase Details\n*******");
		PurchaseDetails purchaseDetails = new PurchaseDetails("P1001","Credit Card");
		System.out.println("Total purchase amount: " +
Math.round(purchaseDetails.calculateTax(100)*100)/100.0);
		System.out.println("Tax percentage: "+purchaseDetails.getTaxPercentage());

		System.out.println("Seller Details\n*******");
		Seller seller = new Seller("Middle east");
		System.out.println("Tax to be paid by the seller: " +
Math.round(seller.calculateTax(100)*100)/100.0);
		System.out.println("Tax percentage: "+seller.getTaxPercentage());
	}

}
```

ASSIGNMENTS

---

1. METHOD OVERRIDING 1

```
class Faculty{
	  //Implement your code here
	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 double calculateSalary(){
		double                                                     facultySalary=
this.getBasicSalary()*(1+(this.getBonusPercentage()/100)+(this.getCarAllowancePercentage()
/100));
		this.setBasicSalary((float) facultySalary);
		return facultySalary;
	}
	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;
	}

}

class OfficeStaff extends Faculty{
	 //Implement your code here
	private String designation;
	public OfficeStaff(String name, float basicSalary, String designation){
		super(name,basicSalary);
		this.designation= designation;
	}
	@Override
public double calculateSalary(){
		double additionalPay =super.calculateSalary();
		double staffSalary=0;
		if(this.getDesignation().equals("Accountant")){
			staffSalary= additionalPay+10000.0;
		}
		else if(this.getDesignation().equals("Clerk")){
			staffSalary= additionalPay+7000.0;
		}
		else if(this.getDesignation().equals("Peon")){
			staffSalary= additionalPay+4500.0;
		}
```

```
        else{
            staffSalary= additionalPay;
        }
        super.setBasicSalary((float) staffSalary);
        return staffSalary;
        }
    public String getDesignation() {
        return designation;
    }
    public void setDesignation(String designation) {
        this.designation = designation;
    }
}

class Teacher extends Faculty{
    //Implement your code here
    private String qualification;
    public Teacher(String name, float basicSalary, String qualification){
        super(name,basicSalary);
        this.qualification= qualification;
    }
    @Override
    public double calculateSalary(){
        double additionalPay =super.calculateSalary();
        double teacherSalary=0;
        if(this.getQualification().equals("Doctoral")){
            teacherSalary= additionalPay+20000.0;
        }
        else if(this.getQualification().equals("Masters")){
            teacherSalary= additionalPay+18000.0;
        }
        else if(this.getQualification().equals("Bachelors")){
            teacherSalary= additionalPay+15500.0;
        }
        else if(this.getQualification().equals("Associate")){
            teacherSalary= additionalPay+10000.0;
        }
        else{
            teacherSalary= additionalPay;
        }
        this.setBasicSalary((float) teacherSalary);
        return teacherSalary;
        }
    public String getQualification() {
        return qualification;
    }
    public void setQualification(String qualification) {
        this.qualification = qualification;
    }

}


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

      }
}
```

----

2. METHOD OVERRDING-2

```
class Event{
      //Implement your code here
      private String eventName;
      private String participantName;
      private double registrationFee;
      public Event(String eventName, String participantName){
            this.eventName= eventName;
            this.participantName= participantName;
      }
      public void registerEvent(){
            if(this.getEventName().equals("Singing")){
                  this.setRegistrationFee(8);
            }
            else if(this.getEventName().equals("Dancing")){
                  this.setRegistrationFee(10);
            }
            else if(this.getEventName().equals("DigitalArt")){
                  this.setRegistrationFee(12);
            }
            else if(this.getEventName().equals("Acting")){
                  this.setRegistrationFee(15);
            }
            else{
                  this.setRegistrationFee(0);
            }
      }
      public String getEventName() {
            return eventName;
      }
      public void setEventName(String eventName) {
            this.eventName = eventName;
      }
      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;
	}

}

class SoloEvent extends Event{
	//Implement your code here
	private int participantNo;
	public SoloEvent(String eventName, String participantName, int participantNo){
		super(eventName, participantName);
		this.participantNo= participantNo;
	}
	@Override
public void registerEvent(){
		super.registerEvent();
	}
public int getParticipantNo() {
	return participantNo;
}
public void setParticipantNo(int participantNo) {
	this.participantNo = participantNo;
}
}

class TeamEvent extends Event{
	//Implement your code here
	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;
	}
	@Override
public void registerEvent(){
		double fee=0;
		if(this.getNoOfParticipants()>1){
		if(this.getEventName().equals("Singing")){
			fee= (double)(this.getNoOfParticipants()*4);
			this.setRegistrationFee(fee);
		}
		else if(this.getEventName().equals("Dancing")){
			fee= (double)(this.getNoOfParticipants()*6);
			this.setRegistrationFee(fee);
		}
		else if(this.getEventName().equals("DigitalArt")){
			fee= (double)(this.getNoOfParticipants()*8);
			this.setRegistrationFee(fee);
		}
		else if(this.getEventName().equals("Acting")){
			fee= (double)(this.getNoOfParticipants()*10);
			this.setRegistrationFee(fee);
		}
		else{
			this.setRegistrationFee(0);
		}
		}
```

```
    }
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;
}

}

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");
        }
    }
}
```

----

3. FINAL 1

```
class Circle{
    private final double PI=3.14;
    private double diameter;
    private double circumference;
    private double area;
    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 double getPI() {
        return PI;
    }

    public Circle(double diameter) {
        this.diameter=diameter;
    }
    public void calculateCircumference() {
        circumference=this.getDiameter()*(this.PI);
        this.setCircumference(circumference);
    }
    public void calculateArea() {
        double r=this.getDiameter()/2;
        area=(this.PI)*(r*r);
        this.setArea(area);
    }
}
 class FinalAssignment {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Circle circle1 = new Circle(12.2);
        Circle circle2 = new Circle(33.2);

        //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();
        }
    }

}
```

----

4. ABSTRACT 1

```
abstract class Payment{
    private int customerId;
```

```
	protected String paymentId;
	protected double serviceTaxPercentage;
	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 Payment(int customerId) {
		this.customerId=customerId;
	}
	public abstract double payBill(double amount);
}

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

	public DebitCardPayment(int customerId) {
		super(customerId);
		paymentId="D"+ ++counter;
		this.setPaymentId(paymentId);
	}

	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) {
		double tax,discount,bill=0.0;
		if(amount<=500) {
			double serviceTaxPercentage=2.5;
			this.setServiceTaxPercentage(serviceTaxPercentage);
			discountPercentage=1;
			discount=(amount*discountPercentage)/100;
			tax=amount+(amount*serviceTaxPercentage/100);
```

```
            bill=tax-discount;
        }
        else if(amount>500 && amount<=1000 ) {
            double serviceTaxPercentage=4;
            this.setServiceTaxPercentage(serviceTaxPercentage);
            discountPercentage=2;
            discount=(amount*discountPercentage)/100;
            tax=amount+(amount*serviceTaxPercentage/100);
            bill=tax-discount;
        }
        else if(amount>1000 ) {
            double serviceTaxPercentage=5;
            this.setServiceTaxPercentage(serviceTaxPercentage);
            discountPercentage=3;
            discount=(amount*discountPercentage)/100;
            tax=amount+(amount*serviceTaxPercentage/100);
            bill=tax-discount;
        }
        return bill;
    }
}

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

    public CreditCardPayment(int customerId) {
        super(customerId);
        paymentId="C"+ ++counter;
        this.setPaymentId(paymentId);
    }

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

    public double payBill(double amount) {
        double total=0.0;
        if(amount<=500) {
            double serviceTaxPercentage=3;
            this.setServiceTaxPercentage(serviceTaxPercentage);
            total=amount+(amount*serviceTaxPercentage/100);
        }
        else if(amount>500 && amount<=1000 ) {
            double serviceTaxPercentage=5;
            this.setServiceTaxPercentage(serviceTaxPercentage);
            total=amount+(amount*serviceTaxPercentage/100);
        }
        else if(amount>1000 ) {
            double serviceTaxPercentage=6;
            this.setServiceTaxPercentage(serviceTaxPercentage);
            total=amount+(amount*serviceTaxPercentage/100);
        }
        return total;
    }
}

 class AbstractAssignment {
```

```
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		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);
	}

}
```
----

5. INTERFACE 1

```
interface Testable{
	boolean testCompatibility();
}

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

class SmartPhone extends Mobile implements Testable{
	private String networkGeneration;

	public String getNetworkGeneration() {
		return networkGeneration;
	}

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

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

	public boolean testCompatibility(){
		if(this.getOperatingSystemName().equals("Saturn")) {

if(this.networkGeneration.equals("3G")&&(this.getOperatingSystemVersion().equals("1.1"))){
				return true;
			}
			else
if(this.networkGeneration.equals("3G")&&(this.getOperatingSystemVersion().equals("1.2"))){
				return true;
			}
			else
if(this.networkGeneration.equals("3G")&&(this.getOperatingSystemVersion().equals("1.3"))){
				return true;
			}
			else
if(this.networkGeneration.equals("4G")&&(this.getOperatingSystemVersion().equals("1.2"))){
				return true;
			}
			else
if(this.networkGeneration.equals("4G")&&(this.getOperatingSystemVersion().equals("1.3"))){
				return true;
			}
			else
if(this.networkGeneration.equals("5G")&&(this.getOperatingSystemVersion().equals("1.3"))){
				return true;
			}
			else {
				return false;
			}

		}
		else if(this.getOperatingSystemName().equals("Gara")) {

if(this.networkGeneration.equals("3G")&&(this.getOperatingSystemVersion().equals("EXRT.1"))
){
                    return true;
                }
                else
if(this.networkGeneration.equals("3G")&&(this.getOperatingSystemVersion().equals("EXRT.2"))
){
                    return true;
                }
                else
if(this.networkGeneration.equals("3G")&&(this.getOperatingSystemVersion().equals("EXRU.1"))
){
                    return true;
                }
                else
if(this.networkGeneration.equals("4G")&&(this.getOperatingSystemVersion().equals("EXRT.2"))
){
                    return true;
                }
                else
if(this.networkGeneration.equals("4G")&&(this.getOperatingSystemVersion().equals("EXRU.1"))
){
                    return true;
                }
                else
if(this.networkGeneration.equals("5G")&&(this.getOperatingSystemVersion().equals("EXRU.1"))
){
                    return true;
                }
                else {
                    return false;
                }

            }
            else
                return false;
        }
}
 class InterfaceAssignment {

        public static void main(String[] args) {
            // TODO Auto-generated method stub
            SmartPhone smartPhone = new SmartPhone("Quick6Pro", "Nebula", "Marco", "1.45",
"4G");
            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!");
        }

}
-------------------------------------
```