Java Assignment

1) Take 20 integer inputs from user and print the following:

number of positive numbers

number of negative numbers

number of odd numbers

number of even numbers

number of 0s.

Open

Answers.java
~/Desktop/java_part-3

```java
import java.util.*;
class Answers {
        public static void main(String args[]) {
                int[] arr=new int[20];
                int pos=0,neg=0,odd=0,even=0,zero=0;
                Scanner sc=new Scanner(System.in);
                for(int i=0; i<20; i++) {
                        arr[i]=sc.nextInt();
                }
                for(int x=0; x<20; x++) {
                        if(arr[x]>=0)
                                pos++;
                        else
                                neg++;

                        if(arr[x]%2==0)
                                even++;
                        else
                                odd++;

                        if(arr[x]==0)
                                zero++;
                }
                System.out.println("Number of positive numbers: "+pos);
                System.out.println("Number of negative numbers: "+neg);
                System.out.println("Number of odd numbers: "+odd);
                System.out.println("Number of even numbers: "+even);
                System.out.println("Number of zeroes: "+zero);
        }
}
```

```
lt-shivamg1@lt-shivamg1: ~/Desktop/java_part-3
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$ javac Answers.java
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$ java Answers
12 23 65 0 7 0 82 44 0 99 1076 56 387 54 83 56 24 58 73 597
Number of positive numbers: 20
Number of negative numbers: 0
Number of odd numbers: 8
Number of even numbers: 12
Number of zeroes: 3
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$
```

2) Take an array of 10 elements. Split it into middle and store the elements in two different arrays. E.g.-

INITIAL array :

58 24 13 15 63 9 8 81 1 78

After splitting :

58 24 13 15 63

9 8 81 1 78

Answers.java
~/Desktop/java_part-3

```java
import java.util.*;
class Answers {
        public static void main(String args[]) {
                int[] arr=new int[10];
                int[] ans1=new int[5];
                int[] ans2=new int[5];
                Scanner sc=new Scanner(System.in);
                for(int i=0; i<10; i++) {
                        arr[i]=sc.nextInt();
                }
                for(int x=0; x<5; x++) {
                        ans1[x]=arr[x];
                        ans2[x]=arr[x+5];
                }
                System.out.println("\nInitial array: ");
                for(int i=0; i<10; i++) {
                        System.out.print(arr[i]+" ");
                }
                System.out.println("\n\nAfter splitting: ");
                for(int i=0; i<5; i++) {
                        System.out.print(ans1[i]+" ");
                }
                System.out.print("\n");
                for(int i=0; i<5; i++) {
                        System.out.print(ans2[i]+" ");
                }
                System.out.print("\n");
        }
}
```

```
lt-shivamg1@lt-shivamg1: ~/Desktop/java_part-3
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$ javac Answers.java
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$ java Answers
12 56 88 439 798 345 8 786 0 97

Initial array:
12 56 88 439 798 345 8 786 0 97

After splitting:
12 56 88 439 798
345 8 786 0 97
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$
```

3) Create a class with a method that prints "This is parent class" and its subclass with another method that prints "This is child class". Now, create an object for each of the class and call

1 - method of parent class by object of parent class

2 - method of child class by object of child class

3 - method of parent class by object of child class

Answers.java
~/Desktop/java_part-3

```java
class Parent {
        public void printP() {
                System.out.println("This is the parent class");
        }
}
class Child extends Parent {
        public void printC() {
                System.out.println("This is the child class");
        }
}
class Answers {
        public static void main(String args[]) {
                Parent p=new Parent();
                Child c=new Child();
                p.printP();
                c.printC();
                c.printP();
        }
}
```

```
lt-shivamg1@lt-shivamg1: ~/Desktop/java_part-3
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$ javac Answers.java
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$ java Answers
This is the parent class
This is the child class
This is the parent class
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$
```

4) Write a program to print the name, salary and date of joining of 10 employees in a company.

Use array of objects.

Answers.java
~/Desktop/java_part-3

```java
class Employee {
        String name;
        int salary;
        String doj;
        Employee(String name,int salary,String doj) {
                this.name=name;
                this.salary=salary;
                this.doj=doj;
        }
        public void print() {
                System.out.println(name+"\t"+salary+"\t"+doj);
        }
}
class Answers {
        public static void main(String args[]) {
                Employee[] data=new Employee[10];
                data[0]=new Employee("Shivam",500000,"27/03/2023");
                data[1]=new Employee("Anil",700000,"30/05/1997");
                data[2]=new Employee("Shubham",467500,"10/12/2001");
                data[3]=new Employee("Rajat",487640,"31/01/2010");
                data[4]=new Employee("Vivek",4396489,"22/02/2000");
                data[5]=new Employee("Rishabh",6764370,"31/01/2010");
                data[6]=new Employee("Nitin",4778780,"04/04/2004");
                data[7]=new Employee("Puneet",6453368,"11/07/1998");
                data[8]=new Employee("Vineet",4519878,"25/08/2005");
                data[9]=new Employee("Sahil",4672863,"22/06/2008");
                System.out.println("Name\tSalary\tDate of Joining");
                for(int x=0; x<10; x++) {
                        System.out.println(data[x].name+"\t"+data[x].salary+"\t"+data[x].doj);
                }
        }
}
```

```
lt-shivamg1@lt-shivamg1: ~/Desktop/java_part-3
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$ javac Answers.java
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$ java Answers
Name    Salary  Date of Joining
Shivam  500000  27/03/2023
Anil    700000  30/05/1997
Shubham 467500  10/12/2001
Rajat   487640  31/01/2010
Vivek   4396489 22/02/2000
Rishabh 6764370 31/01/2010
Nitin   4778780 04/04/2004
Puneet  6453368 11/07/1998
Vineet  4519878 25/08/2005
Sahil   4672863 22/06/2008
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$
```

5) Write a program that takes your full name as input and displays the abbreviations of the first and middle names except the last name which is displayed as it is. For example, if your name is Robert Brett Roser, then the output should be R.B.Roser.

Open

Answers.java
~/Desktop/java_part-3

```java
import java.util.*;
class Answers {
        public static void main(String args[]) {
                Scanner sc=new Scanner(System.in);
                String name=sc.nextLine();
                String[] parts=name.split("\\s");
                for(int i=0; i<2; i++) {
                        Character x=parts[i].charAt(0);
                        parts[i]=x.toString();
                }
                System.out.println(parts[0]+". "+parts[1]+". "+parts[2]);
        }
}
```

```
lt-shivamg1@lt-shivamg1: ~/Desktop/java_part-3
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$ javac Answers.java
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$ java Answers
Shivam Kumar Gupta
S. K. Gupta
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$
```

6) What is the difference between equals() method and equality operator (==) in Java?

Ans. The equals() method compares the values of two objects and returns true if their values are same even if their references are different.

The equality operator(==) compares the references of two objects and returns true only if their references are same. If the references are different, it will return false even if their values are same.

Example:

```
String s1= “Alpha”;

String s2=new String(“Alpha”);

s1==s2; //false

s1.equals(s2); //true
```

7) What is the difference between StringBuilder and StringBuffer?

Ans. StringBuffer is synchronized, i.e., only one thread can work on a StringBuffer object at a particular time.

StringBuilder is non-synchronized, i.e., multiple threads can work on a StringBuilder object at a particular time.

8) Explain the use of final keyword in variable, method and class.

Ans. If a variable is made final, its value cannot be changed.

If a method is made final, it cannot be overridden.

If a class is made final, it cannot be extended.

9) Is it possible that the ‘finally’ block will not be executed? If yes then list the case.

Ans. The ‘finally’ block may not execute if JVM exits while try or catch block is being executed.

```
try {
        System.out.println(“Try block”);
        System.exit(0);
}
finally {
        System.out.println(“Finally block”);
}
```

In this case, the finally block will not execute because JVM has already exited the program before encountering the finally block.

10) What are shallow copy and deep copy in java?

Ans. For an object, if we are making a copy of some entities and some entities are not copied:

In shallow copy, new memory is allocated only to those entities that were copied. The entities that were not copied do not get new memory allocated to them. Only their reference is copied.

In deep copy, new memory is allocated to all entities, whether they were copied or not.

11) What will be the output of below program?

```
public class TestClass
{
   public static void main(String[] args)
   {
      int a = 30;
      int b = 40;
      int c = 10;
      int expression = (a * b)/(a - b + c);
      System.out.println("Result: " +expression);
     }
}
```

Ans. Exception in thread "main" java.lang.ArithmeticException: / by zero at TestClass.main(TestClass.java:9)

12) Why it is always recommended to keep the clean-up activities like closing the I/O resources or DB connections inside a finally block?

Ans. Because the finally block will always execute, irrespective of whether an exception was caught in the try block or not.

13) What happens if the below code is executed?

```
public class Test
```

```
{
    public static void main(String[] args)
    {
        int[] list = new int[4];
        System.out.println(list[4]);
    }
}
```

Ans. Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 4 at Test.main(Test.java:6)

14) How many objects will be created for the following codes:

A.

```
String str1 = "abc";                    //Line1
String str2 = new String("abc");        //Line2
```

Ans. 2 objects

B.

```
String str1 = "abc";                    //Line1
String str2 = "abc";                    //Line2
```

Ans. 1 object

C.

```
String str1 = new String("abc");            //Line1
String str2 = new String("abc");            //Line2
```

Ans. 3 objects

15) How do you check whether a String is empty in Java?

Ans. Using the isEmpty() method of String class. It returns true if String is empty otherwise it returns false.

16) Write a program in java to join two arraylists into one arraylist.

Open

Answers.java
~/Desktop/java_part-3

```java
import java.util.*;
class Answers {
        public static void main(String args[]) {
                ArrayList<Integer> one=new ArrayList<Integer>();
                ArrayList<Integer> two=new ArrayList<Integer>();
                for(int i=10; i<100; i+=10) {
                        one.add(i);
                        two.add(i);
                }
                System.out.println("Before joining:");
                System.out.println(one);
                System.out.println(two);
                two.addAll(one);
                System.out.println("\nAfter joining:");
                System.out.println(two);
        }
}
```

lt-shivamg1@lt-shivamg1: ~/Desktop/java_part-3

```
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$ javac Answers.java
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$ java Answers
Before joining:
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[10, 20, 30, 40, 50, 60, 70, 80, 90]

After joining:
[10, 20, 30, 40, 50, 60, 70, 80, 90, 10, 20, 30, 40, 50, 60, 70, 80, 90]
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$
```

17) Which of the following methods can be used to set every element of the List to a specified value?

set()

add()

complete()

fill()

Ans. The fill() method of Collections class can be used to set every element of the List to a specified value.

18) Which of the following guarantees type-safety in a collection?

Abstract Classes

Interface

Collection

Generics

Ans. Generics guarantee type-safety in a collection. Type-safety is the extent to which a programming language discourages type errors. Type errors are those errors that result from attempts to perform operations on values that are not of the appropriate data type, e.g., adding a string to an integer when there's no definition on how to handle this case. In collections, this is done by specifying the data type of the collection in Generics (<>).

19) Differentiate between Comparable and Comparator in the context of Java.

Ans.

Comparable	**Comparator**
Comparable provides a **single sorting sequence**. In other words, we can sort the collection on the basis of a single element such as id, name, and price.	The Comparator provides **multiple sorting sequences**. In other words, we can sort the collection on the basis of multiple elements such as id, name, and price etc.
Comparable **affects the original class**, i.e., the actual class is modified.	Comparator **doesn't affect the original class**, i.e., the actual class is not modified.
Comparable provides **compareTo() method** to sort elements.	Comparator provides **compare() method** to sort elements.
Comparable is present in **java.lang** package.	A Comparator is present in the **java.util** package.

We can sort the list elements of Comparable type by **Collections.sort(List)** method.	We can sort the list elements of Comparator type by **Collections.sort(List, Comparator)** method.

20) Write a Java program to create and throw custom exceptions.

Open

Answers.java
~/Desktop/java_part-3

```java
class CustomException extends Exception {
        CustomException(String s) {
                System.out.println("Custom exception created");
        }
}
class Answers {
        public static void main(String args[]) {
                try {
                        throw new CustomException("Hello");
                }
                catch(CustomException ex) {
                        System.out.println("Custom exception caught");
                }
                finally {
                        System.out.println("Goodbye");
                }
        }
}
```

lt-shivamg1@lt-shivamg1: ~/Desktop/java_part-3

```
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$ javac Answers.java
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$ java Answers
Custom exception created
Custom exception caught
Goodbye
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$
```

21) What is the output of the below code?

```
class IABC{
  public static void main(String args[])
  {
     String obj = "Hello";
     String obj1 = "ABC";
     String obj2 = "Hello";
     System.out.println(obj.equals(obj1) + " " + obj.equals(obj2));
  }
}
```

Ans. false true

22) Create a class named 'Member' having the following members:

Data members

1 - Name

2 - Age

3 - Phone number

4 - Address

5 - Salary

It also has a method named 'printSalary' which prints the salary of the members.

Two classes 'Employee' and 'Manager' inherits the 'Member' class. The 'Employee' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now, assign name, age, phone number, address and salary to an employee and a manager by making an object of both of these classes and print the same.

Answers.java
~/Desktop/java_part-3

```java
class Member {
        String name,phone,address;
        int age,salary;
        Member(String name,int age,String phone,String address,int salary) {
                this.name=name;  this.age=age;  this.phone=phone;  this.address=address;  this.salary=salary;
        }
        void printSalary() {
                System.out.println("Salary of member "+this.name+" is "+this.salary);
        }
}
class Employee extends Member {
        String specialization;
        Employee(String name,int age,String phone,String address,int salary,String specialization) {
                super(name,age,phone,address,salary);
                this.specialization=specialization;
        }
        void printSalary() {
                System.out.println("Salary of employee "+this.name+" is "+this.salary);
        }
}
class Manager extends Member {
        String department;
        Manager(String name,int age,String phone,String address,int salary,String department) {
                super(name,age,phone,address,salary);
                this.department=department;
        }
        void printSalary() {
                System.out.println("Salary of manager "+this.name+" is "+this.salary);
        }
}
class Answers {
        public static void main(String args[]) {
                Employee e=new Employee("Shivam",22,"6583595485","XYZ Street",500000,"Backend");
                Manager m=new Manager("Amit",30,"3448683547","ABC Street",2000000,"Development");
                e.printSalary();
                m.printSalary();
        }
```

lt-shivamg1@lt-shivamg1: ~/Desktop/java_part-3

```
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$ javac Answers.java
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$ java Answers
Salary of employee Shivam is 500000
Salary of manager Amit is 2000000
lt-shivamg1@lt-shivamg1:~/Desktop/java_part-3$
```