

Rishabh Rajpurohit
Java Part-3 Assignment Solutions

A1.

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

public class Main {
    public static void main(String[] args) {
        System.out.println("Enter 20 Integers: ");
        int[] int_arr = new int[20];
        try (Scanner s = new Scanner(System.in)) {
            for(int i=0;i<20;i++){
                int_arr[i] = s.nextInt();
            }
        }
        catch(Exception e){
            //e.printStackTrace();
            System.out.println("Exception caught while taking input!");
        }
        Q1 q1 = new Q1(int_arr);
        System.out.println("Positive Nums: "+q1.getPositive_numbers());
        System.out.println("Negative Nums: "+q1.getNegative_numbers());
        System.out.println("Odd Nums: "+q1.getOdd_numbers());
        System.out.println("Even Nums: "+q1.getEven_numbers());
        System.out.println("Zeroes: "+q1.getPositive_numbers());
    }
}

class Q1 {
    private List<Integer> int_list = null;
    public List<Integer> getInt_list() {
        return int_list;
    }

    public void setInt_list(List<Integer> int_list) {
        this.int_list = int_list;
    }

    private Integer positive_numbers = 0;
    public Integer getPositive_numbers() {
        return positive_numbers;
    }

    private Integer negative_numbers = 0;
    public Integer getNegative_numbers() {
        return negative_numbers;
    }
}
```

```
private Integer odd_numbers = 0;
public Integer getOdd_numbers() {
    return odd_numbers;
}
```

```
private Integer even_numbers = 0;
public Integer getEven_numbers() {
    return even_numbers;
}
```

```
private Integer zeroes = 0;
```

```
public Integer getZeroes() {
    return zeroes;
}
```

```
Q1(int[] int_arr){
    int_list = new ArrayList<>();
    for(int number:int_arr){
        if(number%2==0) even_numbers++;
        else odd_numbers++;
        if(number>0) positive_numbers++;
        else if(number<0) negative_numbers++;
        else zeroes++;
        int_list.add(number);
    }
}
```

```
}
```

```
lt-rishabhr@lt-rishabhr:~/Docu
otcamp$ java Main
Enter 20 Integers:
-10
-2
4
12
21
0
14
-72
23
0
1123
-20
0
233
32
245
11
-4
131
1
Positive Nums: 12
Negative Nums: 5
Odd Nums: 8
Even Nums: 12
Zeroes: 12
```

A2.

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

public class Main {
    public static void main(String[] args) {
        int size_arr = 10;
        System.out.println("Enter Integers: ");
        int[] int_arr = new int[size_arr];
        try (Scanner s = new Scanner(System.in)) {
            for(int i=0;i<size_arr;i++){
                int_arr[i] = s.nextInt();
            }
        }
        catch(Exception e){
            //e.printStackTrace();
            System.out.println("Exception caught while taking input!");
        }
        Q2 q2 = new Q2(int_arr);
        /*
        System.out.println("Positive Nums: "+q1.getPositive_numbers());
        System.out.println("Negative Nums: "+q1.getNegative_numbers());
        System.out.println("Odd Nums: "+q1.getOdd_numbers());
        System.out.println("Even Nums: "+q1.getEven_numbers());
        System.out.println("Zeroes: "+q1.getPositive_numbers());
        */

        System.out.println("First-half: "+q2.getFirst_half().toString());
        System.out.println("Second-half: "+q2.getSecond_half().toString());
    }
}

class Q1 {
    private List<Integer> int_list = null;
    public List<Integer> getInt_list() {
        return int_list;
    }

    public void setInt_list(List<Integer> int_list) {
        this.int_list = int_list;
    }

    private Integer positive_numbers = 0;
    public Integer getPositive_numbers() {
        return positive_numbers;
    }

    private Integer negative_numbers = 0;
```

```
public Integer getNegative_numbers() {  
    return negative_numbers;  
}
```

```
private Integer odd_numbers = 0;  
public Integer getOdd_numbers() {  
    return odd_numbers;  
}
```

```
private Integer even_numbers = 0;  
public Integer getEven_numbers() {  
    return even_numbers;  
}
```

```
private Integer zeroes = 0;
```

```
public Integer getZeroes() {  
    return zeroes;  
}
```

```
Q1(int[] int_arr){  
    int_list = new ArrayList<>();  
    for(int number:int_arr){  
        if(number%2==0) even_numbers++;  
        else odd_numbers++;  
        if(number>0) positive_numbers++;  
        else if(number<0) negative_numbers++;  
        else zeroes++;  
        int_list.add(number);  
    }  
}
```

```
}
```

```
class Q2 extends Q1{  
    private List<Integer> first_half = null;  
    public List<Integer> getFirst_half() {  
        return first_half;  
    }  
    private List<Integer> second_half = null;  
    public List<Integer> getSecond_half() {  
        return second_half;  
    }  
    Q2(int[] int_arr){  
        super(int_arr);  
        first_half = new ArrayList<>();  
        second_half = new ArrayList<>();  
        int size_array = int_arr.length;
```

```

        for(int i=0;i<size_array;i++){
            if(i<size_array/2) first_half.add(int_arr[i]);
            else second_half.add(int_arr[i]);
        }
    }
}

```

```

● lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$ javac Main.java
● lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$ java Main
Enter Integers:
1
2
3
4
5
6
7
8
9
0
First-half: [1, 2, 3, 4, 5]
Second-half: [6, 7, 8, 9, 0]
○ lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$

```

A3.

```

public class Main {
    public static void main(String[] args) {
        Parent p = new Parent();
        Child c = new Child();
        p.doSomething();
        c.doSomething();
        c.doSomething(true);
    }
}

class Parent{

    public void doSomething(){
        System.out.println("This is parent class.");
    }
}

class Child extends Parent{

    public void doSomething(){
        System.out.println("This is child class.");
    }
}

```

```

    public void doSomething(Boolean callParentMethod){
        super.doSomething();
    }
}

```

```

lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$ javac Main.java
lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$ java Main
This is parent class.
This is child class.
This is parent class.
lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$ 

```

A4.

```

public class Main {
    public static void main(String[] args) {
        Employee[] employees = new Employee[10];
        for(int i=0;i<10;i++) {
            employees[i] = new Employee("Name"+i, i*10000,"Today");
        }

        for(Employee employee: employees){
            System.out.println(employee.toString());
        }
    }
}

```

```

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

    public String getName() {
        return name;
    }

    private Integer salary;
    public void setSalary(Integer salary) {
        this.salary = salary;
    }

    public Integer getSalary() {
        return salary;
    }
}

```

```

private String doj;

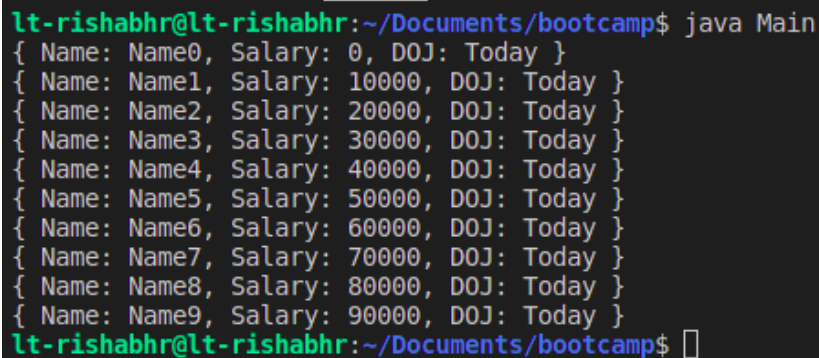
public void setDoj(String doj) {
    this.doj = doj;
}

public String getDoj() {
    return doj;
}

Employee(String name, Integer salary, String doj){
    this.name = name;
    this.salary = salary;
    this.doj = doj;
}

public String toString(){
    return String.format("{ Name: %s, Salary: %d, DOJ: %s }",this.getName(),this.getSalary(),this.getDoj());
}
}

```



```

lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$ java Main
{ Name: Name0, Salary: 0, DOJ: Today }
{ Name: Name1, Salary: 10000, DOJ: Today }
{ Name: Name2, Salary: 20000, DOJ: Today }
{ Name: Name3, Salary: 30000, DOJ: Today }
{ Name: Name4, Salary: 40000, DOJ: Today }
{ Name: Name5, Salary: 50000, DOJ: Today }
{ Name: Name6, Salary: 60000, DOJ: Today }
{ Name: Name7, Salary: 70000, DOJ: Today }
{ Name: Name8, Salary: 80000, DOJ: Today }
{ Name: Name9, Salary: 90000, DOJ: Today }
lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$ 

```

A5.

```

public class Main {
    public static void main(String[] args) {
        Person p = new Person("Neil Nitin Mukesh");
        System.out.println("Name: "+p.getName());
        System.out.println("Shortname: "+p.getShortName());
    }
}

class Person{
    private String name;

```

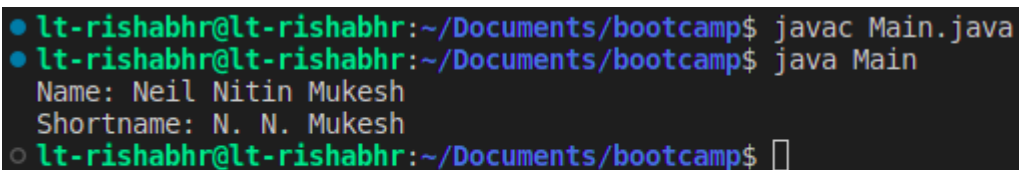
```

public String getName() {
    return name;
}

Person(String name){
    this.name = name;
}

public String getShortName(){
    String[] words = name.split("\\s+");
    String temp = words[0].charAt(0)+" "+words[1].charAt(0)+" "+words[2];
    return temp;
}
}

```



```

● lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$ javac Main.java
● lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$ java Main
Name: Neil Nitin Mukesh
Shortname: N. N. Mukesh
○ lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$ █

```

A6.

(==) this operator compares the references of two variables. If two variables point to same object in memory(heap) then it returns true otherwise false. Whereas 'equals' methods compares the actual contents of two objects. For example if two Objects of Person class contains same 'name' and also their references then equals return true. Equals method should be overridden in case of user made objects and should implement 'comparable' interface as a good practice.

A7.

String Buffer has same methods as String Builder except that StringBuffer is synchronized or that it is thread safe. Whereas StringBuilder is not. StringBuffer objects are generally safe to use in multithreading environment, where many threads may be trying to access the same Stringbuffre object at the same time. StringBuffer is slower. than StringBuilder.

A8.

Final Keyword can be used for a variable to declare it as a constant. For ex.

```
Final int a = 5;
```

```
a=6 // this line will give error because we can;t change a final variable's value.
```

Similarly Final Keyword when used with a class, the class is then said to be a constant. And it is done so no other class can inherit it and override its methods.

Similarly final methods of a base class cannot be overridden even when there is a child class.

A9.

Yes there are some cases when the finally method fails to run successfully. Like:

- > if the JVM runs out of memory
- > if our java process is killed by outside processes like task manager
- > if our machine shuts down
- > when the system,exit() method is called in the try block
- > if there is a deadlock condition in our try block

A10.

In primitive data types, shallow copy and deep copy are the same, that is value of variable is copied.

In Objects, when we shallow copy an object, the new object is created with the same reference as the object which is copied. Here the reference is copied. Whereas in a deep copy, a whole new object is created with all the fields same as the object which is copied. Here the reference of object is not copied instead its contents.

A11.

Arithmetic exception will cause the program to crash.Because of dividing by zero.

A12.

Because unless there is a hardware failure or outside process kills our java process, the finally block will always run irrespective of unexpected exceptions.

A13.

ArrayIndexOutOfBoundsException will cause program to crash. Because array has size of 4 and we are trying to access the fifth value of array.

A14.

A- 2

B- 1

C- 2

A15.

To check is a string is empty we use the .isEmpty() method. It will return true if it is Empty.

A16.

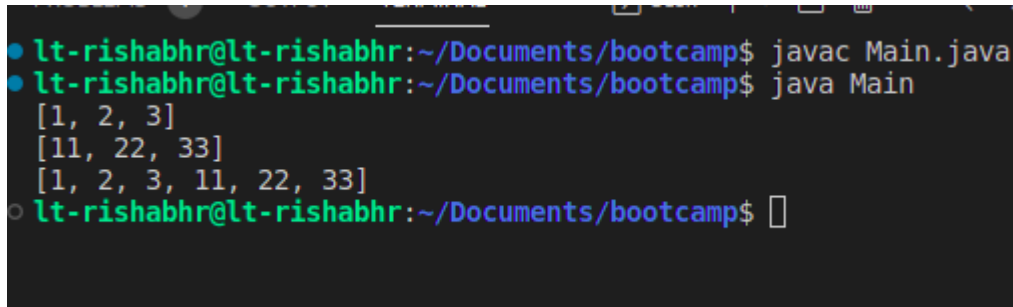
```
import java.util.ArrayList;
```

```
public class Main {  
    public static void main(String[] args) {  
        ArrayList<Integer> a1 = new ArrayList<>();  
        a1.add(1);  
        a1.add(2);  
        a1.add(3);  
        ArrayList<Integer> a2 = new ArrayList<>();  
        a2.add(11);  
    }  
}
```

```

    a2.add(22);
    a2.add(33);
    ArrayList<Integer> a3 = new ArrayList<>(a1);
    a3.addAll(a2);
    System.out.println(a1.toString());
    System.out.println(a2.toString());
    System.out.println(a3.toString());
}
}

```



```

lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$ javac Main.java
lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$ java Main
[1, 2, 3]
[11, 22, 33]
[1, 2, 3, 11, 22, 33]
lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$ 

```

A17.

Fill() method

A18.

Generics

A19.

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.

2) 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.

3) Comparable provides compareTo() method to sort elements. Comparator provides compare() method to sort elements.

4) Comparable is present in java.lang package. A Comparator is present in the java.util package.

5) 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.

A20.

```

public class Main {
    public static void main(String[] args) {
        try {
            throw new MyException();
        } catch (MyException m){
            System.out.println("Rishabh threw an exception.");
            System.out.println(m.getMessage());
        }
    }
}

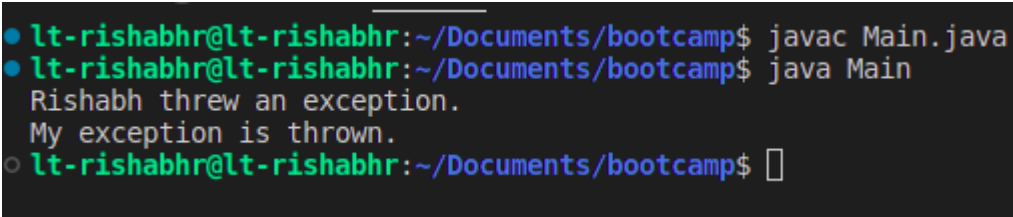
```

```

    }
}

class MyException extends Exception{
    MyException(){
        super("My exception is thrown.");
    }
}

```



```

● lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$ javac Main.java
● lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$ java Main
Rishabh threw an exception.
My exception is thrown.
○ lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$ 

```

A21.

False true

A22.

```

public class Main {
    public static void main(String[] args) {
        Member a = new Employee("Peter B.", "42", "4374327493", "Delhi", "12000", "WEB DEV");
        Member b = new Manager("Tim C.", "55", "4374327493", "Mumbai", "18000", "HR");
        System.out.println(a.toString());
        System.out.println(b.toString());
    }
}

```

```

class Member {
    private String name;

    Member(String name, String age, String phone, String address, String salary) {
        this.name = name;
        this.age = age;
        this.phone = phone;
        this.address = address;
        this.salary = salary;
    }

    public String getName() {
        return name;
    }

    private String age;

    public String getAge() {
        return age;
    }
}

```

```

private String phone;

public String getPhone() {
    return phone;
}

private String address;

public String getAddress() {
    return address;
}

private String salary;

public String getSalary() {
    return salary;
}

public void printSalary() {
    System.out.println("My Salary is: " + this.getSalary());
}

public String toString() {
    return String.format("{ name: %s, age: %s, address: %s, phone: %s, salary: %s", this.getName(),
this.getAge(),
        this.getAddress(), this.getPhone(), this.getSalary());
}
}

class Employee extends Member {
    private String specialization;

    public String getSpecialization() {
        return specialization;
    }

    Employee(String name, String age, String phone, String address, String salary, String specialization)
    {
        super(name, age, phone, address, salary);
        this.specialization = specialization;
    }

    public String toString(){
        String temp = super.toString();
        temp = temp + ", Specialization: " + this.getSpecialization();
        return temp;
    }
}

```

```
}
```

```
class Manager extends Member {  
    private String department;
```

```
    public String getDepartment() {  
        return department;  
    }  
}
```

```
    Manager(String name, String age, String phone, String address, String salary, String department) {  
        super(name, age, phone, address, salary);  
        this.department = department;  
    }  
}
```

```
    public String toString(){  
        String temp = super.toString();  
        temp = temp + ", Department: " + this.getDepartment();  
        return temp;  
    }  
}
```

```
}
```

```
● lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$ java Main  
{ name: Peter B., age: 42, address: Delhi, phone: 4374327493, salary: 12000, Specialization: WEB DEV  
{ name: Tim C., age: 55, address: Mumbai, phone: 4374327493, salary: 18000, Department: HR  
○ lt-rishabhr@lt-rishabhr:~/Documents/bootcamp$
```