ASSIGNMENT

OBJECTIVE

1) B(ArithmeticException).
2) B(start exception finally end).
3) A(car maruti tata).
4) A(MyException).
5) A(Parent p=new Parent();
   Parent.Child c=p.new Child();).
6) A(Compilation error).
7) D(Range).
8) A(tr:nth-child(even){
   Background-color:
   #f2f2f2;
   }).
9) B(init,service,destroy).
10) B(hover).

SUBJECTIVE

1)

Arraylist:

- In first place array is a fixed size. It means, we will declare the number of elements we use at the time of declaring the array. Once we declare the array we cannot change. So, if we want to insert or delete an element from the array, we use the concept called Dynamic array. By using the concept of dynamic array we can change the size of the array by insertion ,deletion etc.

for remove an element from specific index::

```
if (count > 0)   {
for (int i = index; i < count - 1; i++)
{
  array[i] = array[i + 1];
```

```
         }
           array[count - 1] = 0;
         count--;
         }
         }
```

for inserting an element::

```
         Int a[] = {1,2,3};
  System.out.println("Array:"+Arrays.toString(a));

  output:[1,2,3]


      ArrayList<Integer> arrayList = new ArrayList<Integer>();
          arrayList.add(4);
            arrayList.add(5);
        a = arrayList.toArray(a);
            System.out.println("Array after inserting element: "+Arrays.toString(a));

 output:  [1,2,3,4,5]
```

Arraylist consists of duplicate values. That means if we give one value more than one time ,it will give the same number of times.

```
Eg:List lst = new ArrayList<>(2);
                           lst.add(0);
                           lst.add(1);     o/p:[0,1,2,1,4,5]
                           lst.add(2);
                           lst.add(1);
                           lst.add(4);
                           lst.add(5);
                     System.out.println(lst);
```

We also can access values using get method by passing index as a parameter.

```
Eg:List lst = new ArrayList<>(2);
                           lst.add(0);
```

```
                    lst.add(1);     o/p: 1
                    lst.add(2);
                    lst.add(1);
                    lst.add(4);
                    lst.add(5);
               System.out.println(lst.get(3));
```

Linkedlist:

Java LinkedList class can contain duplicate elements.

```
Ex::int main()
{
   Node* head = NULL;
   insert(&head, 5);
   insert(&head, 7);              o/p:2
   insert(&head, 5);
   insert(&head, 1);
   insert(&head, 7);

   cout << countNode(head);

   return 0;
}
```

linked list means one element is connected to another element. So, the linked list class can behaves as both list and queue.As it implements the interfaces of dequeue and lists.

```
Ex:: Queue<Integer> q= new LinkedList<>();
      for (int i = 0; i < 5; i++)
         q.add(i);
   System.out.println("Elements of queue "+ q);
```

2)

Error page means ,When we see a "404 error" or "page not found" on a page ,where we wrote a program for the appearence of webpage on the back.The page says that the user reached the domain requested, but the URL provided with no information.If we want to create an error page ,we have to set page directive attribute **isErrorPage** value to true.

```
<html>
<body>
<%@ page isErrorPage="true" %>
 error:
<%= exception %>
</body>
</html>
```

3)

There are Three types of tags are there:-

1.scriptlet tag

E.g.

```
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

2.expression tag

E.g.

```
 <html>
<body>
Current                    Time:                    <%=
java.util.Calendar.getInstance().getTime() %>
</body>
</html>
```

3.declaration tag

```
<html>
<body>
<%! int data=50; %>
<%= "Value of the variable is:"+data %>
<body>
</html>
```

4)

HTTP is called as a **stateless protocol** because each request is executed independently, without any knowledge of the requests that were executed before it, which means once the transaction ends the connection between the browser and the server is also lost.

5)

These are the ways to iterate over list:

1 .Simple For loop.
2.Enhanced For loop.
3.Iterator.
4.ListIterator.
5.While loop.
6.Iterable.forEach() util.
7.Stream.forEach() util.

The Fail Fast iterators immediately throw ConcurrentModificationException in case of structural modification of the collection. Structural modification means adding, removing, updating the value of an element in a data collection while another thread is iterating over that collection.

OverCome:-

The Fail Safe iterators are just opposite to Fail Fast iterators; unlike them, A fail-safe iterator does not throw any exceptions unless it can handle if the collection is modified during the iteration process. This can be done because they operate on the copy of the collection object instead of the original object. The structural changes performed on the original collection ignored by them and affect the copied collection.

6)

- **getSession()** : Returns the current session associated with this request, or if the request does not have a session, creates one.
- **getSession(true)** : Returns the current HttpSession associated with this request, if there is no current session, returns a new session
- **getSession(false)** : Returns the current HttpSession associated with this request, if there is no current session, returns null.
- getSession() is preferable because it checks and creates the new one.

7)

a Java **memory leak happens when an application unintentionally (due to logical errors in code) holds on to object references that are no longer required.** These unintentional object references prevent the built-in Java garbage collection mechanism from freeing up the memory consumed by these objects.

We can prevent memory leaks by

- Release the session when it is no longer needed. Use the HttpSession.invalidate() to do this.
- Keep the time-out time low for each session.
- Store only the necessary data in your HttpSession.
- Avoid using string concatenation. Use StringBuffer's append() method because the string is an unchangeable object while

string concatenation creates many unnecessary objects. A large number of temporary objects will slow down performance.

- As much as possible, you should not create HttpSession on your jsp page. You can do this by using the page directive <%@page session=”false”%>.
- If you are writing a frequently executed query, use PreparedStatement object rather than using Statement object. Why? PreparedStatement is precompiled, while Statement is compiled every time your SQL statement is transmitted to the database.
- When using JDBC code, avoid using “*” when you write your query. Try to use the corresponding column name instead.
- If you are going to use stmt = con.prepareStatement(sql query) within a loop, then be sure to close it inside that particular loop.
- Be sure to close the Statement and ResultSet when you need to reuse these.
- Close the ResultSet, Connection, PreparedStatement and Statement in the final block.

8)

- **Yes, we can declare a class as final**.
- Final is a non-access modifier.
- when we declare a class as a final it is known as final class.
- the final class can’t be inherited.
- There are two uses of the final class

  1) One is to prevent **inheritence**

  Eg:

```
final class A{
   Int tyres;
   Int gears;
  }
Class B extends A{
```

```
Int brake;
}
Output:compile error
```

2) The second use of final with classes is to **create an immutable class** like the predefined String class.you can't make a class immutable without making it final.

9)

In **multi-threading,** access to the resources which are shared among multiple threads can be controlled by using the concept of **synchronization.** Using synchronized keyword, we can ensure that only one thread can use shared resource at a time and others can get control of the resource only once it has become free from the other one using it.

10)

Clone in Java refers to **the creation of an Exact copy of an object**. It creates a new instance of the class of the Existing object and initializes all its attributes with exact Values of the corresponding attributes of this object. there is no operator to create a copy of an object.we have a method to create the clone of the object called **Clone()** method.

The **java.lang.Cloneable** interface must be implemented by the class whose object is going to create .if we don't implement the **cloneable** interface it generates an exception "**CloneNotsupportedException"**.

If We use Clone() method it saves the extra processing task for creating the exact copy of an object.if we use new keyword for creating object it takes a lot of processing time that's why we use object cloning method.

Eg;

```
class Student implements Cloneable{
     int rollno;
  String name;
Student(int rollno,String name){
```

```
               this.rollno=rollno;
                 this.name=name;
                   }
 public Object clone()throws CloneNotSupportedException{
return super.clone();
}
public static void main(String args[]){
try{
Student t1=new Student(1,"Ravi");

Student t2=(Student)t1.clone();

System.out.println(t1.rollno+" "+t1.name);
System.out.println(t2.rollno+" "+t2.name);

}catch(CloneNotSupportedException c){}

}
}
Output: 1 Ravi
        1 Ravi
```

<u>PROGRAMMING</u>

1)

```
        import java.util.*;
        public class Main{
        public static void main(String[] args) {
       Scanner s = new Scanner(System.in);
       int count=0;
       System.out.println("enter String1");
       String str1=s.next();
       System.out.println("enter String2");
       String str2=s.next();
       boolean result=str1.contains(str2);
       if(result){
          System.out.println("true"+    "    "+    "(string1    contain
string2)");
       }
       else{
          System.out.println("false"+  "  "+  "(string1  not  contain
string2)");
       }

    }
}
```

2)

```
     import java.util.Arrays;
      import java.util.Scanner;
    public class MainProgram
      {
   public static void main(String args[])
        {
     Scanner sc = new Scanner(System.in);
     int count=0;
     System.out.println("Enter the size of the array that is to be
created: ");
     int size = sc.nextInt();
     int[] myArray = new int[size];
     System.out.println("Enter the elements of the array: ");
     for(int i=0; i<size; i++){
       myArray[i] = sc.nextInt();
     }
     System.out.println("Enter the number: ");
     int num = sc.nextInt();
    System.out.println("The array created is: 
"+Arrays.toString(myArray));
    System.out.println("indices of the elements whose sum is: 
"+num);
    for(int i=0; i<myArray.length; i++){
      for (int j=i+1; j<myArray.length; j++){
        if((myArray[i]+myArray[j])== num){
          count++;
          System.out.println(myArray[i]+","+myArray[j]);
          System.out.println(myArray[j]+","+myArray[i]);
        }
        }

        }
        if(count==0){
            System.out.println("cannot get value for given 
sum");
          }
        }
      }
```

---------END--------