Programs on Exception Handling in Java

---

1. What kind of checked or unchecked exception will the following programs throw, if any?

a)

```
public class Test

{

public static void main(String[] args)

{

 int x = 10;

 int y = 0;

 int z = x / y;

 System.out.println(z);

  }

}
```

Output: ArithmeticException

b)

```
public class Test

{

public static void main(String[] args)

{
```

```
double n[] = {3.2, 3, 4, 6};

int a = 4;

double x = n[0]/(a - n[2]);

System.out.println(x);

}

}
```

Output: Infinity

c)

```
public class Test

{

public static void main(String[] args)

{

String str = "Scientech";

System.out.println(str.charAt(9));

  }

}
```

Output: StringIndexOutOfBoundsException

d)

```
public class Test

{
```

```
public static void main(String[] args)
{
 Object obj = new Object();
 String str = (String)obj;
 System.out.println(str);
  }
}
```

Output: ClassCastException

e)

```
public class Test
{
public static void main(String[] args)
{
 System.out.println("Hello");
 Object obj = null;
 System.out.println(obj.toString());
  }
}
```

Output: NullPointerException

f)

```
public class Test

{

public static void main(String[] args)

{

 System.out.println("Hello Java");

 int[] list = {1, 2, 3, 4, 5 };

 System.out.println(list[5]);

  }

}
```

Output: ArrayIndexOutOfBoundsException

g)

```
public class Test

{

public static void main(String[] args)

{

  String str = "Java Programming";

  int a = Integer.parseInt(str);

  System.out.println("Value of a: " +a);

  }
```

```
}
```

Output: NumberFormatException

h)

```
public class Test

{

public static void main(String[] args)

{

 byte num;

 System.out.println("Scientech");

  num = Byte.parseByte("Easy");

  System.out.println("num: " +num);

  }

}
```

Output: Scientech, NumberFormatException

i)

```
public class Test

{

public static void main(String[] args)

{
```

```
 for(int i = 1; i <= 5; i++)

  System.out.println("Value of i: " +i);

  Thread.sleep(1000);

  System.out.println("Hello Java");

  }

}
```

Ouput: InterruptedException

j)

```
import java.io.File;

import java.io.FileInputStream;

public class Test

{

public static void main(String[] args)

{

 File file = new File("not_existing_file.txt");

 FileInputStream stream = new FileInputStream(file);

  }

}
```

Output: FileNotFoundException

2. What will be the output of the following program when run?

```
public class Test
```

```
{
public static void main(String[] args)
{
try {
 int value = 5;
 if (value < 10)
    throw new RuntimeException("Value is less than 10");
 }
catch (RuntimeException re) {
   System.out.println(re.getMessage());
  }
System.out.println("Outside try-catch block");
  }
}
```

Output: Value is less than 10, Outside try-catch block

3. What would be the output in the program 2 if the line int value = 5; is changed to int value = 20;?

Output: Outside try-catch block

4. What would be the output of the following programs?

a)

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

```
{
for (int i = 1; i <= 5; i++) {
 System.out.print(i + " ");
try {
 int a = 10;
 int b = 0;
 int c = a/b;
 System.out.println(c);
 }
catch (ArithmeticException ae) {
 }
 }
}}
```

Output: 1 2 3 4 5

b)

```
class Test {
public static void main(String[] args)
{
try {
 for (int i = 1; i <= 5; i++) {
```

```
 System.out.print(i + " ");

 int a = 20;

 int b = 0;

 int c = a/b;

 System.out.println(c);

 }

}

catch (ArithmeticException ae) {

 }

}

}
```

Output: 1

c)

```
public class Test {

public static void main(String[] args)

{

try {

 for (int i = 1; i <= 5; i++)

 {

  System.out.print(i + " ");
```

```
 }
}
catch (Exception e) {
 System.out.println(e.getMessage());
 }
 System.out.println("\nOut of try-catch block");
 }
}
```

Output: 1 2 3 4 5 , Out of try-catch block

d)

```
class Test {
public static void main(String[] args)
{
 System.out.println("111");
try
{
 int x = 12/0;
 System.out.println("Result of x: " +x);
 System.out.println("333");
 }
```

```
catch(ArithmeticException ae)

{

  System.out.println("Hello world");

 }

 System.out.println("444");

 }

}
```

Output: 111, Hello world, 444

e)

```
class Test {

int x = 30, y = 0;

void divide()

{

 System.out.println("I am in method");

try

{

 System.out.println("I am in try block");

 int z = x/y;

 System.out.println("Result of z: " +z);

}
```

```
catch(NullPointerException np)

{

  System.out.println("I am in catch block");

 }

}

public static void main(String[] args)

{

 Test t = new Test();

 t.divide();

 }

}
```

Output: I am in method, I am in try block, Exception in thread “main” java.lang.ArithmeticException: / by zero

f)

```
class Test {

public static void main(String[] args)

{

System.out.println("111");

try

{

 System.out.println("222");
```

```
 double y = 1.0/0;

 }

catch(ArithmeticException e)

{

try

{

 System.out.println("Hello");

  int x = 20/0;

 }

catch(NullPointerException np)

{

 System.out.println("333");

 }

}

System.out.println("444");

 }

}
```

Output: 111, 222, 444

g)

```
class Test {
```

```
public static void main(String[] args)
{
System.out.println("111");
try
{
 System.out.println("222");
 int y = 1/0;
 }
catch(ArithmeticException e)
{
try
{
 System.out.println("Hello");
  double x = 2.5/0;
  System.out.println("Java");
 }
catch(NullPointerException np)
{
 System.out.println("333");
 }
```

```
}
System.out.println("444");
 }
}
```

Output: 111, 222, Hello, Java, 444

h)

```
class Test {
public static void main(String[] args)
{
System.out.println("111");
try
{
 System.out.println("222");
 }
catch(ArithmeticException e)
{
try
{
  double x = 2/0;
 }
```

```
catch(NullPointerException np)

{

 }

}

System.out.println("ABC");

 }

}
```

Output: 111, 222, ABC

5. What exception will be thrown when the following program is run?

```
class Test {

public static void main(String[] args)

{

try

{

 int[] list = new int[5];

 System.out.println(list[5]);

 }

catch(StringIndexOutOfBoundsException e)

{

try {
```

```
 System.out.println("ABC");

 int x = 10/0;

}

catch(ArithmeticException ae){

 }

}

System.out.println("ABC");

 }

}
```

Output: ArrayIndexOutOfBoundsException

6. What exception will be thrown when the program is run?

```
class Test {

public static void main(String[] args)

{

try {

  int[] list = new int[10];

  System.out.println("list[10] is " + list[10]);

 }

catch (ArithmeticException ex) {

  System.out.println("ArithmeticException");
```

```
 }
catch (RuntimeException ex) {
  System.out.println("RuntimeException");
 }
catch (Exception ex) {
  System.out.println("Exception");
 }
 }
}
```

Output: RuntimeException

7. What is displayed on the console when the following program is run?

a)

```
class Test {
public static void main(String[] args)
{
try {
  method();
  System.out.println("After the method call");
 }
catch (StringIndexOutOfBoundsException se) {
```

```
  System.out.println("StringIndexOutOfBoundsException");
}
catch (RuntimeException ex) {
  System.out.println("RuntimeException");
}
catch (Exception e) {
  System.out.println("Exception");
 }
}
static void method() throws Exception {
String str = "Hello";
char ch = str.charAt(5);
System.out.println(ch);
 }
}
```

Output: StringIndexOutOfBoundsException

b)

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

```
try {
  method();
 System.out.println("I am in try block");
 }
catch (RuntimeException ex) {
  System.out.println("RuntimeException in main");
}
catch (Exception ex) {
  System.out.println("Exception in main");
  }
}
static void method() throws Exception {
try {
  int a[] = {10, 20, 30, 40};
  a[10] = 5;
}
catch (ArrayIndexOutOfBoundsException ae) {
 System.out.println("ArrayIndexOutOfBoundsException in method()");
}
catch (RuntimeException re) {
```

```
  System.out.println("RuntimeException in method()");
}
catch (Exception e) {
  System.out.println("Exception in method()");
  }
 }
}
```

Output: ArrayIndexOutOfBoundsException in method(), I am in try block

c)

```
class Test {
public static void main(String[] args)
{
try {
  method();
 System.out.println("I am in try block");
 }
catch(Exception e){
 System.out.println("I am in catch block");
}
finally {
```

```
  System.out.println("I am in finally block");
 }
}
static void method() throws Exception {
try {
  int a[] = {10, 20, 30, 40};
  a[4] = 20;
  System.out.println(a[4]);

try {
   int x = 1/0;
   System.out.println(x);
  }
catch(ArithmeticException ae){
   System.out.println("I am in inner catch block");
  }
}
catch (ArrayIndexOutOfBoundsException ae) {
 System.out.println("I am in outer catch block");
}
```

```
 }

}
```

Output: I am in outer catch block, I am in try block, I am in finally block

d)

```
class Test {

public static void main(String[] args){

try {

  method();

 System.out.println("I am in try block");

 int a = 1/0;

 }

catch(Exception e){

 System.out.println("I am in catch block");

}

System.out.println("I am in main method");

}

static void method() throws Exception {

try {

  int a[] = {10, 20, 30, 40};

  a[3] = 20;
```

```
  System.out.println(a[3]);



try {

  int x = 1/0;

  System.out.println(x);

  }

catch(ArithmeticException ae){

  System.out.println("I am in inner catch block");

  }

}

catch (ArrayIndexOutOfBoundsException ae) {

 System.out.println("I am in outer catch block");

}

 }}
```

Output: 20, I am in inner catch block, I am in try block, I am in catch block, I am in main method

e)

```
class Test {

public static void main(String[] args) throws Exception{

try {

  method();
```

```
 }
finally{
 System.out.println("I am in finally block");
}
}
static void method() throws Exception {
try {
int[]  list = new int[5];
System.out.println(list[0]);
try {
  System.out.println(list[5]);
  }
catch(ArithmeticException ae){
   System.out.println("I am in inner catch block");
  }
}
catch (ArrayIndexOutOfBoundsException ae) {
 System.out.println("I am in outer catch block");
}
 }}
```

Output: 0, I am in outer catch block, I am in finally block

8. What will be the output of the program when an exception occurred in try block?

```
class Test {

public static void main(String[] args) throws Exception

{

try {

 System.out.println("ABC");

 int x = 1/0;

 System.out.println("PQR");

try {

 System.out.println("XYZ");

  }

catch(ArithmeticException ae) {

  System.out.println("I am in inner catch block");

  }

}

catch (ArrayIndexOutOfBoundsException ae) {

 System.out.println("I am in outer catch block");

   }

 }
```

```
}
```

Output: ABC, Exception in thread “main” java.lang.ArithmeticException: / by zero.

9. What will be the output when the following program has complied?

```
class Test {

public static void main(String[] args) throws Exception

{

try {

 System.out.println("ABC");

 int a  = 5, b = 10, c = 5;

 a += 5;

 b -= a + c;

 int x = (a + b)/(b + c);

 System.out.println(x);

System.exit(0);

}

catch(ArithmeticException ae){

 System.out.println("PQR");

}

finally{
```

```
 System.out.println("XYZ");

}

 }

}
```

Output: ABC, PQR, XYZ

10. What will be the output of the program when System.exit(0); statement is used just above an exception in try block?

```
class Test {

public static void main(String[] args) throws Exception

{

try {

 System.out.println("ABC");

 System.exit(0);

 int x = 1 / 0;

 }

catch(ArithmeticException ae){

 System.out.println("PQR");

}

finally {

   System.out.println("XYZ");

  }
```

```
 }

}
```

Output: ABC

11. Will finally block be executed in the below program? What will be the output?

```
class Test {

public static void main(String[] args) throws Exception

{

try {

 System.out.println("ABC");

 int x = 1 / 0;

 }

catch(ArithmeticException ae){

 System.out.println("PQR");

 System.exit(0);

}

finally {

   System.out.println("XYZ");

  }

 }

}
```

Output: No, finally block will not be executed.

ABC, PQR

12. Will the program be successfully compiled?

```
class Test {

int m1(){

try {

  System.out.println("ABC");

  return 50;

 }

catch(Exception e)

{

  System.out.println("I am in catch block");

 }

}

public static void main(String[] args)

{

Test t = new Test();

t.m1();

 }

}
```

Ans: No, Compile time error.

13. What will be displayed on the console when the following program is run?

a)

```
class Test {

int m1() {

try {

  System.out.println("ABC");

  return 50;

 }

catch(Exception e)

{

  System.out.println("I am in catch block");

 }

return 10;

}

public static void main(String[] args)

{

Test t = new Test();

System.out.println(t.m1());

 }
```

```
}
```

Output: ABC, 50

b)

```
class Test {
int m1(){
try {
  System.out.println("ABC");
  int x = 1/0;
  return 50;
 }
catch(ArithmeticException ae)
{
  System.out.println("I am in catch block");
 }
return 10;
}
public static void main(String[] args)
{
Test t = new Test();
System.out.println(t.m1());
```

```
 }
}
```

Output: ABC, I am in catch block, 10

c)

```
class Test {
int m1(){
try {
  System.out.println("ABC");
  int x = 1/0;
  return 50;
 }
catch(ArithmeticException ae)
{
  System.out.println("I am in catch block");
  return 40;
 }
finally{
   return 60;
 }
}
```

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

{

Test t = new Test();

System.out.println(t.m1());

 }

}
```

Output: ABC, I am in catch block, 60

Java Collections Framework Interview Questions

---

1. What is Collection in Java?
Answer: A collection is a group of objects. In Java, these objects are known as elements of the collection.

For example, In childhood, we had a kiddy bank. In the kiddy bank, we had collected a lot of coins. This kiddy bank is called collection and the coins are nothing but objects.

Technically, a collection is a container object that stores a group of other objects as a single unit or single entity.

2. Does a collection object store copies of other objects?
Answer: No, a collection object works with reference types. It never stores copies of other objects. It stores references of other objects.

3. Can we store a primitive data type into a collection?
Answer: No, collections store only objects, not primitive data types values.

4. Which types of objects can be stored in a collection or container object?

Answer: We can store both homogeneous and heterogeneous objects in the collection or container object.

5. What are duplicate objects in Java?
Answer: The multiple objects of a class that contains the same data are called duplicate objects in java.

---

6. What are unique objects in java?
Answer: The multiple objects of a class that contains different data are called unique objects in java.

7. Why do we need Collections in Java?
Or, what is the use of Collections in Java?

Answer: Collections in Java can be used for storing multiple homogeneous and heterogeneous, duplicate, and unique elements without any size limitation.

8. What is Framework in Java?
Answer: A set of several classes and interfaces which provide a ready-made architecture is called framework in java.

9. What is Collections Framework in Java?
Answer: Collections framework in Java is a sophisticated hierarchy of numerous predefined interfaces and implementation classes that can be used to handle a group of objects as a single entity.

10. In which java package is the collections framework placed?
Answer: The collections framework is placed in java.util package.

---

11. What is the main difference between Collection and Collections Framework in Java?

Answer: Collection in java is an object that has multiple elements of the same type in a single entity. These multiple elements are accessed through one Collection object.

Whereas, Collections Framework is a library of interfaces and implementation classes that provides the common architecture for creating, accessing, and updating the different types of collections.

12. What are the different components of collections framework?
Answer: The different components of collections framework are interfaces, implementation classes, and algorithms.

13. What is the difference between Arrays and Collections in Java?
Or, Explain the difference between Collections and Arrays.
Answer: The main difference between arrays and collections are as follows:

a) Arrays are fixed in size/length but collections are growable in nature. We can increase or decrease size based on our requirements.

b) Arrays are not recommended to use with respect to memory whereas collections are recommended to use with respect to memory.

c) Arrays are recommended to use with respect to performance but collections are not recommended to use with respect to performance.

d) Arrays can store only homogeneous data elements (similar type of data) but collections can hold both homogeneous and heterogeneous elements.

e) Arrays do not support any method whereas collections support numerous types of methods.

g) Arrays can hold both primitives and object types whereas, collections can hold only objects but not primitive.

---

14. What are the main benefits/advantages of Collections Framework in Java?

Answer: The main benefits/advantages of Collections Framework in Java are as follows:

a) **Reusability:** Collections Framework in Java provides common classes and utility methods that can be used with various types of collections. It enables the reusability of the code.
b) **Quality:** Collections Framework improves the quality of code since the code is already tested and used by thousands of programmers.
c) **Speed:** Collections Framework improves the performance of the applications, reduces the development time and the burden of designers, programmers, and users.
d) **Maintenance:** Since Java Collections framework code is open source and API documents are widely available, therefore, the code is easier to maintain written with the help of the collections framework.
e) **Growable:** The size of the collection container is growable in nature.

15. What are the limitations of Collections Framework in Java?

Answer: There are two limitations of collections framework in Java. They are as follows:

- Care must be taken to use the appropriate cast operation.
- Compile type checking is not possible.

---

16. What is the root interface of collection hierarchy in Java?

Answer: The root interface of collection hierarchy in Java is collection interface. Since collection interface extends Iterable interface, therefore, some people consider Iterable interface as the root interface.

17. In which Java package Iterable interface is present?

Answer: Iterable interface is present in java.lang package. Whereas, Collection interface is present in java.util package.

18. Explain the collection hierarchy in Java with a diagram.

Answer: Go to this tutorial: *Collection Hierarchy in Java*

19. Which method of iterable interface is used to iterate over elements of the collection?

Answer: Iterable interface provides only one method called iterator() method.

20. Which interfaces extend the collection interface in java?

Or, What are the basic interfaces available in Java collections framework?

Answer: List, Queue, and Set are three interfaces that extend the Collection interface. The map interface is not inherited by collection interface.

The list of basic interfaces available in Java collections framework are as follows:

- Collection interface
- List interface
- Set interface
- Queue interface
- Map interface

---

21. What are the key differences between Collection and Collections?

Answer: The key differences between Collection and Collections are as follows:

a) In Java, Collection is an interface whereas, Collections is a class.

b) Collection is a base interface whereas, Collections is a utility class.

c) Collection provides numerous methods that are used for data structures that contain objects. Whereas, Collections provides many methods that are used for operations such as access, find, etc. on a collection.

22. What are several Thread-safe classes provided by Java Collections framework?

Answer: Several Thread-safe classes in Java Collections framework are as follows:

- Stack
- Properties
- Vector
- Hashtable
- BlockingQueue
- ConcurrentMap
- ConcurrentNavigableMap

23. What is the difference between collections and streams in java?

Answer: The major differences between collections and streams in java are as follows:

a) Collections hold an array of values whereas, streams do not hold any values.

b)

24. What are the two ways to traverse elements of a collection in java?

Answer: We can traverse elements of a collection either by using for-each or Iterator.

---

25. Suppose that we are traversing elements of a collection using an Iterator. Can we remove elements from a collection using Iterator? If yes, how to?

Answer: Iterator provides remove() method that removes the last element that is returned by the next method.

For example, the following code snippet removes "John" from the List using Iterator.

```
List<String> list = new ArrayList<String>();
```

```
  list.add("Ivaan");

  list.add("John");

  list.add("Merry");



Iterator<String> itr = list.iterator();

while(itr.hasNext())

{

   String element = itr.next();

   if(element.equals("John"))

   {

      element.remove();

    }

}
```

26. Is it possible to convert a collection into an array?

Answer: Collection interface provides toArray() method to convert a collection to an array. A sample of code is given below:

```
Object[ ] obj = list.toArray();
```

27. Suppose we have a collection of String type and want to convey it into an Array of String type. How can it be achieved?

Answer: list.toArray() will always return an array of Object type. To Array of type String, the following code snippet can be used:

```
String[ ] myArray = myStringList.toArray(new String[0]);
```

28. Which of Collection classes implement Set interface in java?

Answer: There are several collection classes in java that implement Set interface. They are:

- AbstractSet
- ConcurrentSkipListSet
- CopyOnWriteArraySet
- EnumSet
- HashSet
- LinkedHashSet
- TreeSet

29. What are Collection classes that implement List interface in java?

Answer: Collection classes that implement List interface, are:

- AbstractList
- AbstractSequentialList
- ArrayList
- AttributeList
- CopyOnWriteArrayList
- LinkedList
- RoleList
- RoleUnresolvedList
- Stack
- Vector

30. What are Java collection classes that implement Queue interface?

Answer: There are the following collection classes that implement Queue interface in java, are as:

- LinkedList
- PriorityQueue
- ArrayQueue
- PriorityBlockingQueue
- LinkedBlockingQueue

31. Which collection interface extends Queue interface in java?

Answer: Deque interface extends Queue interface. It was added to the collection framework in Java SE6.

32. What are concrete subclasses that implement the Deque interface?
Answer: The concrete classes such as LinkedList and ArrayDeque implement Deque interface.

33. What are concrete classes that implement Map interface in java?
Answer: The concrete classes that implement Map interface, are as follows:

- HashMap
- HashTable
- LinkedHashMap
- TreeMap

34. Which interface extends Map interface and is also implemented by TreeMap class?
Answer: SortedMap interface extends Map interface.

35. Which method is used to get the total number of elements in a collection?
Answer: Collection interface provides size() method that can be used to get elements of a collection.

---

36. Is it a good approach to use Generic features in collections?
Or, **What is the key advantage of the generic collection?**
Answer: Yes, it is a good approach to use generics in collections because generics provide type safety. While iterating elements of a collection, typecasting is not required if uses generic class.

For more detail, go to this tutorial: *Generics in Java*.

37. What are the bulk operations supported by collection interface in java?

Answer: The collection interface supports the following bulk operations such as: add, addAll, clear, containsAll, removeAll, retainAll, etc.

38. How to check a collection is empty or not?
Answer: Collection interface provides a method isEmpty() that can be used to check a collection is empty or not.

39. Is it possible that a collection object can be cloned and serialized?
Answer: Yes, it is possible to clone and serialize a collection object because all the concrete classes in the Java Collections Framework implement the java.lang.Cloneable and java.io.Serializable interfaces, except java.util.PriorityQueue that does not implement the Cloneable interface.

40. What method will you use to add all the elements from one collection to another collection?
Answer: addAll(Collection c).

---

41. Why Map interface does not extend Collection interface in Java?
Answer: Map interface and its implementation classes are the part of Java collections framework. But Map is not collection and collection is not map. Map stores elements in key-value pair fashion. Hence, it does not make any sense for Map to extend Collection interface or vice-versa.

42. What are the legacy classes that are not a part of collections framework now?
Answer: The legacy classes that are not part of collections framework, are as follows:

- Vector
- HashTable
- Dictionary
- Properties
- Stack

43. What is List in Java?

Answer: A list in Java is a collection for storing elements in sequential order. It is a sub-interface of the collection interface that is available in java.util package.

44. What are the concrete subclasses that implement List interface?

Answer: The concrete subclasses that implement List interface, are: ArrayList, LinkedList, Vector, and Stack.

45. What are the important features of list in Java?

Answer: The most important features of list in java are as follows:

- List is an ordered collection where elements are maintained by index positions because it uses an index-based structure.
- It allows storing duplicate elements in Java.
- We can add an element at any position in the list.
- List maintains insertion order.
- It allows for storing many null elements.
- List uses a resizable array for its implementation. It means we can increase or decrease the size of the array.
- Using ListIterator, we can iterate elements of list in both directions.

---

46. How to create an object of generic list in Java?

Answer: After the introduction of Generics in Java 1.5, we can restrict the type of object that can be stored in the list.

The general syntax to create a list of objects with a generic type parameter is as:

```
List<T> list = new ArrayList<T>(); // Here, T is a type parameter.
```

For example:

```
a. List<String> list = new ArrayList<String>(); // Creating a list of objects of String type using ArrayList.
```

47. What is the result of the following code snippet?

```
import java.util.ArrayList;

import java.util.List;

public class Test

{

  public static void main(String[] args)

  {

     List<String> list = new ArrayList<>();

      list.add("Monday");

      list.add("Tuesday");

      list.remove(0);

      System.out.println(list.get(0));

   }

}
```

Answer: Output: Tuesday

48. Will the below code snippet compile fine? If yes, what will be the result of the following program?

```
import java.util.ArrayList;

import java.util.List;

public class Addition
```

```
{
 public static void main(String[] args)
 {
   List<String> list = new ArrayList<String>();
    list.add("Herry");
    list.add("Deep");
    list.stream().forEach(s -> System.out.println(s));
  }
}
```

Answer: Yes, the above code will compile successfully. Output: Herry, Deep.

49. What will be the result of the following program?

```
import java.util.ArrayList;
import java.util.List;
public class Test {
public static void main(String[] args)
{
  List<String> list = new ArrayList<String>();
   list.add("Dhanbad");
   list.add("Cape town");
   list.add("New York");
```

```
  list.removeIf( x -> x.length() > 10);

  System.out.println(list.size());

 }

}
```

Answer: Output: 3

50. Is it possible to traverse a list in both directions?

Answer: Yes, we can traverse a list in both directions using ListIterator.

51. How do we create a list from an array of objects?

Answer: Arrays class provides asList() method that can be used to create a list from an array of objects. The sample of code is as follows:

```
import java.util.Arrays;

import java.util.List;

public class Test {

public static void main(String[] args)

{

   String[ ] str = {"Dhanbad", "Ranchi", "Mumbai"};

   List<String> list = Arrays.asList(str);

   System.out.println(list);

  }

}

Output:

          [Dhanbad, Ranchi, Mumbai]
```

Test Your Knowledge

---

52. Suppose that list1 is a list that contains strings Apple, Orange, and Banana and that list2 is another list that contains the strings Apple, Orange, and Guava.

Answer the following questions:

a. What will be the result of list1 and list2 after executing list1.addAll(list2)?

b. What will be the result of list1 and list2 after executing list1.add(list2)?

c. What will be the result of list1 and list2 after executing list1.removeAll(list2)?

d. What will be the result of list1 and list2 after executing list1.remove(list2)?

e. What will be the result of list1 and list2 after executing list1.retainAll(list2)?

f. What will be the result of list1 after executing list1.clear()?