Collections

1. Find Output

```
import java.util.*;

public class priorityQueue
{
   public static void main(String[] args)
   {
      PriorityQueue<Integer> queue =
                        new PriorityQueue<>();
      queue.add(11);
      queue.add(10);
      queue.add(22);
      queue.add(5);
      queue.add(12);
      queue.add(2);

      while (queue.isEmpty() == false)
         System.out.printf("%d ", queue.remove());

      System.out.println("\n");
   }
}
```

Output

a) 11 10 22 5 12 2
b) 2 12 5 22 10 11
c) 2 5 10 11 12 22
d) 22 12 11 10 5 2
Ans. (c)
Explanation: Priority queue always outputs the minimum element from the queue when remove() method is called, no matter what the sequence of input is.

2. Find Output

```
import java.util.*;

public class Test
{
   public static void main(String[] args)
   {
      TreeSet<String> treeSet = new TreeSet<>();

      treeSet.add("raju");
      treeSet.add("anil");
      treeSet.add("hari");
      treeSet.add("babu");

      for (String temp : treeSet)
         System.out.printf(temp + " ");

      System.out.println("\n");
   }
}
```

Output

a) anil babu hari raju
b) raju anil hari babu
c) raju hari babu anil
d) Exception

Ans) a

Explanation: A TreeSet sorts the data in ascending order that is inserted in it. Therefore, the output string contains all the strings arranged in ascending order.

3. Find Output

```
import java.util.*;

public class Test
{
   public static void main(String[] args)
   {
      TreeSet<String> treeSet = new TreeSet<>();

      treeSet.add("raju");
      treeSet.add("10");
      treeSet.add("45.5");

      for (String temp : treeSet)
         System.out.printf(temp + " ");

      System.out.println("\n");
   }
}
```

Output

a) 10 45.5 raju
b) raju 10 45.5
c) 45.5 10 raju
d) Exception

Ans: a

4. Find Output

```
import java.util.*;

public class Test
{
   public static void main(String[] args)
   {
      TreeSet<String> treeSet = new TreeSet<>();

      treeSet.add("raju");
      treeSet.add(10);
      treeSet.add("45.5");

      for (String temp : treeSet)
         System.out.printf(temp + " ");

      System.out.println("\n");
   }
}
```

Output

a) Compilation error
b) raju 10 45.5
c) 45.5 10 raju
d) Exception

Ans: a

5. Find Output

```
import java.util.*;

public class Test
{
   public static void main(String[] args)
   {
      List<String> list1 = new LinkedList<>();
      list1.add("raju");
      list1.add("for");
      list1.add("raju");
      list1.add("rani");
      list1.add("java");

      List<String> list2 = new LinkedList<>();
      list2.add("raju");

      list1.removeAll(list2);

      for (String temp : list1)
         System.out.printf(temp + " ");

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

Output

a) for rani java

b) raju for rani java

c) Compilation error

d) raju

Ans) a

Explanation: list1.removeAll(list2) function deletes all the occurance of string in list2 from list1.

6. Find Output

```
import java.util.*;

public class Test
{
   public static void main(String[] args)
   {
      HashSet<String> hashSet = new HashSet<>();
      hashSet.add("java");
      hashSet.add("for");
      hashSet.add("java");
      hashSet.add("java program");

      System.out.println(hashSet);

   }
}
```

Output

a) [java program, for, java]
b) [java program, for, java, java]
c) Compilation error
d) Exception

Ans: a

Explanation: A HashSet is a set and as a set doesn't contain any duplicate element therefore, the string 'java' appears only once in the output.

7. Find Output

```
import java.util.*;

public class Test
{
   public static void main(String[] args)
   {
      List<String> list = new LinkedList<>();
      list.add("10");
      list.add("20");
      list.add("5");
      list.add("2");
      Iterator<Integer> iter = list.iterator();

      while (iter.hasNext())
         System.out.printf(iter.next() + " ");

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

Output

a) Runtime Error
b) Compilation Error

c) 10 20 50 2

d) 2 50 20 10

Ans. b

Explanation: Instead of “Iterator<Integer> iter = list.iterator();”, we have to use “Iterator<String> iter = list.iterator();”

An iterator made for iterating over Integer cannot be used to iterate over String data type.

8. Find Output

```
import java.util.ArrayList;
class Demo {
public void show()
   {
      ArrayList<Integer> list = new ArrayList<Integer>();
      list.add(4);
      list.add(7);
      list.add(1);
      for (int number : list) {
         System.out.print(number + " ");
      }
   }
} public class Main {
public static void main(String[] args)
   {
      Demo demo = new Demo();
      demo.show();
   }
}
```

Output

A. Compilation Error
B. 4 7 1
C. 1 4 7
D. None
Answer : B

Explanation : List in java stores its elements in Sequential manner it maintains insertion order. List provides the ability of accessing elements using index. Collections are in the package util so we are importing java.util.ArrayList.

9. Find Output

```
import java.util.LinkedList;

class Demo {
public void show()
   {
      LinkedList<String> list = new LinkedList<String>();
      list.add("Element1"); // line 6
      list.add("Element2");
      System.out.print(list.getFirst()); // line 8
   }
} public class Main {
public static void main(String[] args)
   {
      Demo demo = new Demo();
      demo.show();
   }
}
```

Output

A. Element1
B. Compilation Error at line 8
C. Runtime Error

D. Element2

Ans) A

Explanation : LinkedList has a getFirst() method . It returns an elements at Zero index. LinkedList also maintains its insertion order and provides easy accessing of elements.

10. Find Output

```
import java.util.ArrayList;
class Demo {
public void show()
   {
      ArrayList<String> list = new ArrayList<String>();
      System.out.print(list.get(0));
   }
} public class Main {
public static void main(String[] args)
   {
      Demo demo = new Demo();
      demo.show();
   }
}
```

Output

A. ArrayIndexOutOfBoundException
B. IndexOutOfBoundException
C. null
D. “”

Ans: B

Explanation : There is no element present in that index ‘0’ so it is IndexOutOfBoundException.

11. Find Output

```
import java.util.ArrayList;

class Demo1 {
public void show()
   {
      ArrayList<String> list = new ArrayList<String>();
      list.add("java"); // line 6
      list.add("cpp");
      System.out.print(list.getFirst()); // line 8
   }
} public class Test {
public static void main(String[] args)
   {
      Demo1 demo = new Demo1();
      demo.show();
   }
}
```

Output

A. java
B. Compilation Error
C. Runtime Error

D. cpp

Answer: B

Explanation: ArrayList doesn’t have method getFirst(). So it is compilation error.getmethod() is available only LinkedList. Therefore, it provide compilation error in this program.

12. Find Output

```
import java.util.LinkedList;

class Demo {
public void show()
   {
      LinkedList<String> list = new LinkedList<String>();

      System.out.print(list.getFirst());
   }
} public class Main {
public static void main(String[] args)
   {
      Demo demo = new Demo();
      demo.show();
   }
}
```

Output

A. null
B. IndexOutOfBoundException
C. NoSuchElementException

D. Compilation error

Answer: C

Explanation: There is no element in LinkedList so it return NoSuchElementException. NoSuchElementException is a RuntimeException thrown when there is no more element in it. NoSuchElementException extends RuntimeException.

13. Find Output

```
import java.util.LinkedList;

class Demo {
public void show()
   {
      LinkedList<Integer> list = new LinkedList<Integer>();
      list.add(1);
      list.add(4);
      list.add(7);
      list.add(5);
      for (int i = 0; i < list.size(); i++) {
         System.out.print(list.get(i) + " ");
      }
   }
}
public class Main {
public static void main(String[] args)
   {
      Demo demo = new Demo();
      demo.show();
   }
}
```

Output

A. Compilation Error
B. 1 4 7 5
C. 1 4 5 7

D. exception

Answer: B

Explanation: List stores element in sequential order and then we can access element in List using index. List provides the ability to access its elements by using its index. But in set, map elements are not accessed by using index.

14. Find Output

```
import java.util.Collections;
import java.util.LinkedList;
import java.util.List;

class Demo {
public void show()
   {
      List<Integer> list = new LinkedList<Integer>();
      list.add(1);
      list.add(4);
      list.add(7);
      list.add(5);
      Collections.sort(list); // line 9
      System.out.println(list);
   }
} public class Main {
public static void main(String[] args)
   {
      Demo demo = new Demo();
      demo.show();
   }
}
```

Output

A. Compilation Error at line 9
B. [1, 4, 5, 7]
C. [1, 4, 7, 5]

D. exception

Answer: B

Explanation: Collections.sort() sort the list in ascending order. Collections class provides static methods for sorting the elements in collections. If Collection elements are of set type elements are inserted in sorted order no need to sort.

15. Find Output

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;

class Demo {
public void show()
   {
      ArrayList<String> list = new ArrayList<String>();
      list.add("banana");
      list.add("apple");
      Iterator itr = list.iterator();

      Collections.sort(list);
      while (itr.hasNext()) {
         System.out.print(itr.next() + " ");
      }
   }
}
public class Main {
public static void main(String[] args)
   {
      Demo demo = new Demo();
      demo.show();
   }
}
```

Output

A. Compilation Error cannot give Collections.sort() after Iterator.
B. apple banana
C. banana apple

D. Exception

Answer: B.

Explanation: Collections.sort() sort element and Iterator is an object used to traverse through a Collection. Iterator is an interface available in Collection framework in java.util.package. It is used to traverse elements one by one.

16. Find Output

```
import java.util.ArrayList;

class Demo {
public void show()
   {
      ArrayList<String> list = new ArrayList<String>();
      ArrayList<Integer> list1 = new ArrayList<Integer>();
      boolean check = (list.getClass() == list1.getClass());
      System.out.println(check);
   }
}
public class Main {
public static void main(String[] args)
   {
      Demo demo = new Demo();
      demo.show();
   }
}
```

Output

A. true
B. false

C. Compilation error

D. Exception

Answer: A

Explanation: getclass() method return the runtime class of an object. That class object is the object that is locked by static synchronized method of represented class. Here both are in ArrayList Class so answer is true.

17. Find Output

```
import java.util.LinkedList;

class Demo {
public void show()
   {
      LinkedList<String> list = new LinkedList<String>();

      System.out.println(list.getClass());
   }
}
public class Main {
public static void main(String[] args)
   {
      Demo demo = new Demo();
      demo.show();
   }
}
```

Output

A. class java.util.LinkedList
B. String
C. Compiler Error

D. Exception

Answer: A

Explanation: getclass() method returns the runtime class of an object. That class object is the object that is locked by static synchronized method of represented class. Here LinkedList is the runtime class so the answer is java.util.LinkedList.

18. Find Output

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
public class Test {
public static void main(String[] args) {
List<String> al = new ArrayList<String>();
al.add("a");
al.add("b");
Iterator itr = al.iterator();
while(itr.hasNext())
{
System.out.print(itr.next());
al.add("c");
}
}
}
```

Output

a) Exception
b) Compilation error
c) abc
d) abccc

Ans) a

19. Find Output

```
import java.util.Arrays;
import java.util.Comparator;
class Sort implements Comparator<Integer>
{
public int compare(Integer o1, Integer o2) {
return o2.compareTo(o1);
}

}

public class Test {
public static void main(String[] args) {
Integer i[] = {2,5,10,1,4};
Arrays.sort(i,new Sort());
for(int i1:i)
System.out.print(i1+" ");
}

}
```

Output

a) 10 5 4 2 1

b) 1 2 4 5 10

c) Compilation error

d) 2 5 10 1 4

Ans) a

20. Find Output

```
import java.util.Iterator;
import java.util.Set;
import java.util.TreeSet;
public class Test {
public static void main(String[] args) {
Set set = new TreeSet();
set.add(1);
set.add("2");
set.add(3);
Iterator itr = set.iterator();
while(itr.hasNext())
{
System.out.println(itr.next());
}

}
}
```

Output

a) ClassCastException
b) StringIndexOutOfBoundsException
c) TreeSetException
d) Exception

Ans) a

21. Find Output

```
import java.util.Iterator;
import java.util.Set;
import java.util.TreeSet;
public class Test {
public static void main(String[] args) {
Set<String> set = new TreeSet<String>();
set.add("Raju");
set.add("Rani");
set.add("Raju");
set.add("rani");
set.add("Anu");
Iterator<String> itr = set.iterator();
while(itr.hasNext())
{
System.out.print(itr.next()+" ");
}

}
}
```

Output

a) Anu Raju Rani rani
b) Anu Raju Raju Rani rani
c) rani Anu Raju Rani
d) rani Anu Raju Raju Rani

Ans) a

22. Find Output

```
import java.util.Collections;
import java.util.Enumeration;
import java.util.Vector;
public class Test {
public static void main(String[] args) {
Vector<String> v = new Vector<String>();
v.add("1");
v.add("2");
Enumeration<String> e = Collections.enumeration(v);
while(e.hasMoreElements())
{
v.add("3");
System.out.print(e.nextElement());
}
}
}
```

Output

a) ConcurrentModificationException
b) ConcurrentException
c) 123
d) 1233

Ans: a

Find Output

Output