Collection Framework Workbook

Answer the Following

1. What does `get(int index)` method define by `List` interface do?_____
2. Collection stores only_____
3. `TreeSet` maintains_____order.
4. What is the data structure that a `Set` uses to store its elements?_____
5. What type of `Map` is used with synchronized access?_____
6. Which method do we need to implement the `java.lang.Comparable` interface?_____
7. How many ways does the `java.lang.Comparable` interface allow us to sort a collection?_____
8. What is returned from both the `compare() and compareTo()` methods?_____
9. Name the collection interface implemented by the `Vector` class_____
10. What is the name of collection interface used to maintain unique elements_____
11. Name the collection interface implemented by the `HashSet` class_____
12. Which collection class allows you to access its elements by associating a key with an element's value, and provides synchronization?_____

State whether the following are True/False

1. The elements in the collection `java.util.HashSet` are guaranteed to be unique. [ ]
2. The `ListIterator` interface extends both the `List` and `Iterator` interfaces. [ ]
3. The `ListIterator` interface provides forward and backward iteration capabilities.[ ]

4. The elements in the collection `java.util.ArrayList` are ordered.[ ]
5. `TreeSet` stores elements sorted ascending order. [ ]
6. `ArrayList` is not synchronized. [ ]
7. `HashMap` is synchronized. [ ]
8. `Set` allows at most one null element. [ ]
9. There are no direct implementation of the `Collection<E>` interface. [ ]
10. If two arrays hold the same elements they are considered equal by the `Arrays.equals()` method. [ ]
11. `Map` interface is derived from the `Collection` interface. [ ]

Multiple Choice Questions

1. Which of these packages contain all the collection classes?
   (a) java.lang
   (b) java.util
   (c) java.net
   (d) java.awt
2. Which interface does `java.util.Hashtable` implement?
   (a) java.util.Map
   (b) java.util.List
   (c) java.util.HashTable
   (d) java.util.Collection
3. Which interface provides the capability to store objects using a key-value pair?
   (a) java.util.Map
   (b) java.util.Set
   (c) java.util.List
   (d) java.util.Collection
4. Which of these classes is not part of collection framework?
   (a) Map
   (b) Array

(c) Stack

(d) Queue

5. Which of these interface is not a part of collection framework?

   (a) List

   (b) Set

   (c) SortedMap

   (d) SortedList

6. Which of the following method is used to return a `Set` that contains the entries in a `Map`?

   (a) keySet()

   (b) getSet()

   (c) entrySet()

   (d) getAll()

7. Which interface is used to traverse a list in both forward and backword direction?

   (a) Iterator

   (b) ListIterator

   (c) Enumeration

   (d) None of these

8. Which of these is Basic interface that all other interfaces inherits?

   (a) Set

   (b) Array

   (c) List

   (d) Collection

9. Which of these classes implement the collection interface `SortedMap`.

   (a) HashMap

   (b) Hashtable

   (c) TreeMap

   (d) HashSet

10. What type of collection would we use if we wanted no duplicates?

    (a) List

(b) Map

(c) Queue

(d) Set

11. What type of List is used with synchronized access?

    (a) ArrayList<E>

    (b) Vector<E>

    (c) LinkedList<E>

12. What type of collection does not extend the Collection<E> interface?

    (a) List

    (b) Map

    (c) Queue

    (d) Set

13. What type of collections are Lists?

    (a) ordered

    (b) sorted

    (c) unordered

14. Which List class is synchronized?

    (a) ArrayList

    (b) LinkedList

    (c) Vector

15. What type of variables can we use with the `java.util.Collections` Class?

    (a) primitive variables

    (b) reference variables

    (c) neither

    (d) both

16. Which interface would we use to sort a class that can't be modified?

    (a) java.lang.Comparable

    (b) java.util.Comparator

Exercises

- Write the expected output, or compiler errors if any, for each of the following programs in the box provided below each program.
- Then execute the programs and check your answers.
- Then answer the questions given below.

Program 1

```
1  import java.util.HashSet;
2  import java.util.Set;
3  import java.util.TreeSet;
4  public class Test{
5      public static void main(String[] args) {
6          Set s = new HashSet();
7          s.add(''A''); // Line 1
8          s.add(new Foo()); // Line 2
9          Set t = new TreeSet();
10         t.add(''A''); // Line 3
11         t.add(new Foo()); // Line 4
12     }
13 }
14 class Foo {
15
16 }
```

Q1: The code will throw a runtime exception, which line causes the exception?

Program 2

```
1  import java.util.SortedSet;
2  import java.util.TreeSet;
3  public class Test {
4      public static void main(String[] args) {
5          TreeSet<Integer> map = new TreeSet<Integer>();
6          map.add(1);
7          map.add(2);
8          map.add(4);
9          map.add(7);
```

```
10          SortedSet<Integer> smap = map.subSet(2,7);
11          map.add(5);
12          map.add(9);
13          System.out.println(smap);
14      }
15  }
```

Q1: What is the result of compiling and running the code?

Program 3

```
1  import java.util.*;
2  public class Test {
3      public static void main(String[] args) {
4          Queue<Integer> queue = new LinkedList<Integer>();
5          queue.add(1);
6          queue.add(3);
7          queue.add(4);
8          queue.add(7);
9          // insert code here
10         System.out.println(queue);
11     }
12 }
```

Q1: What can be inserted, independently, at line // insert code here to cause the program to print out : [3, 4, 7, 0]

Program 4

```
1  import java.util.HashSet;
2  import java.util.Set;
3  public class Test{
4      public static void main(String[] args) {
5          Set<Human> humans = new HashSet<Human>();
6          humans.add(new Human(13));
7          humans.add(new Human(33));
8          humans.add(new Human(21));
9          humans.add(new Human(21));
```

```
10          System.out.print(humans.size()+" ");
11          System.out.print(humans);
12      }
13  }
14  class Human implements Comparable<Human> {
15      Integer age;
16      public Human(int age) {
17          this.age = age;
18      }
19      public int compareTo(Human h) {
20          return h.age.compareTo(this.age);
21      }
22      public String toString() {
23          return "'"+this.age;
24      }
25  }
```

Q1: What is the result of compiling and running the code?

Program 5

```
1  import java.util.*;
2  class Test {
3      public static void main (String[] args) {
4          Object i = new ArrayList().iterator();
5          System.out.print((i instanceof List)+",");
6          System.out.print((i instanceof Iterator)+",");
7          System.out.print(i instanceof ListIterator);
8      }
9  }
```

Q1: What will be the output of the program?

Program 6

```
1  public static void main(String[] args) {
2      Object obj = new Object() {
3          public int hashCode() {
```

```
4                 return 42;
5             }
6         };
7         System.out.println(obj.hashCode());
8     }
```

Q1: What will be the output of the program?

Program 7

```
1  import java.util.*;
2  public class Test {
3      public static void main(String[] args) {
4          Integer int = new Integer(10);
5          Vector vec = new Vector();
6          LinkedList list = new LinkedList();
7          vec.add(int);
8          list.add(int);
9          if(vec.equals(list))
10             System.out.println(``equal'');
11         else
12             System.out.println(``not equal'');
13                                                    }
14     }
15 }
```

Q1: What is the result of attempting to compile and run the code?

Program 8

```
1  import java.util.*;
2  public class Test {
3      public static void main(String[] args) {
4          Integer num1 = new Integer(4);
5          Integer num2 = new Integer(8);
6          Integer num3 = new Integer(4);
7          HashSet hs = new HashSet();
8          hs.add(num1);
```

```
9               hs.add(num2);
10              hs.add(num3);
11              System.out.println(hs);
12          }
13  }
```

Q1: What is the expected output of the above code?