Cognizant Technology Solutions

Java Collections Exercise

CATP Java Team
5/20/2012

For The Associates:

The documents details two flavors of problem statements

- **Statement # 1:** Few problem solutions have been provided for associates should analyze the program and write down the program output. This will enhance the analyzing skills of associates and also understand "why" part of java programming feature. The associates can then try running the program in eclipse and check if the output with what they have written.
- **Stamen # 2:** There are some problem statements provided similar to the final assessment and associates need to solve it. This will enhance the programming skills of the associates.
  IMPORTANT: These exercises will gear you up for the core java assessment so please develop/analyze the exercise independently. In case you are stuck up reach out to the trainers.

Exercises:

1. Declare suitable collection at the position //insert code here

```
class CollectionTypes {

        public static void main(String[ ]  args) {

                        x.add("one");

                        x.add("two");

                        x.add("one");

                         System.out.println(x.poll());

                }

        }
```

Solution=>

```
import java.util.*;
public class Main {

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

```
Queue<String> x = new LinkedList<String>();


                x.add("one");

                x.add("two");

                x.add("one");

                System.out.println(x.poll());
                System.out.println(x);

            }


    }
```

2. What is the result of compiling and running the following program?

```
                public class Tester {

        public static void main(String[] args) {        List<String>
list1 = new ArrayList<String>();//line 1
                List<Object> list2 = list1;//line 2

        list2.add(new Integer(12));//line 3      System.out.println(list2.size());//line 4

                }

                }
```

Solution=>

incompatible types because one List is string type and another is Object Type. So List1 can not be converted to List or we can not assign list1 reference to list2 reference.

3. What is the result of compiling and running the following program?

```
import java.util.*;

public class TestGenericConversion {

public static void main(String s[ ]){

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

list.add("one");

list.add(2);

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

} }
```

Solution=>The above List can only accept string so we can not insert int data type into list. Int cannot be converted to string. So it will give error.

4. What is the result of attempting to compile and run the following code?

```
public class Test {
        public static void main(String[] args){
         Integer a = new Integer(4);
         Integer b = new Integer(8);
         Integer c = new Integer(4);
HashSet hs = new HashSet();
hs.add(a);          hs.add(b);
hs.add(c);
         System.out.println(hs);
        }
    }
```

Solution-4
Output-[4,8]

Explaination: Because HashSet implements the Set interface that does not allow duplicate value So hashset will not added duplicate value and print only 4 and 8;

5.Create a class with a method which can remove all the elements from a list other than the collection of elements specified.

Class Name	ListManager
Method Name	removeElements
Method Description	Remove all the elements from a list other than the collection of elements specified.
Argument	List<String> list1, List<String> list2;
Return Type	List- ArrayList contains the resulting List after the removal process.
Logic	Accept two List objects list1 and list2 and remove all the elements from list 1 other than the elements contained in list2.This should be done in single step process without using loop.

Solution=>

```
import java.util.*;
public class ListManager
{
        public static void main(String[] args) {
           List<String> list1=new ArrayList<String>();
           list1.add("Mango");
      list1.add("Apple");
      list1.add("Banana");
      list1.add("Grapes");
                System.out.println("this is the first ArrayList= "+list1);
                List<String> list2=new ArrayList<String>();
           list2.add("Mango");
      list2.add("Apple");
      list2.add("Grapes");
      list2.add("Tomato");
                System.out.println("this is the second ArrayList= "+list2);
           list1.removeAll(list2);
                System.out.println("Arraylist after removing the element= "+list1);


        }
}
```

5. Create a class that can accept an array of String objects and return them as a sorted List

Class Name	ListManager
Method Name	getArrayList
Method Description	Converts the String array to ArrayList and sorts it
Argument	String []elements
Return Type	List- ArrayList containing the elements of the String array in sorted order
Logic	Load the elements in to an ArrayList and sort it.

Solution=>

```
import java.util.*;
public class Main
{
        public static void main(String[] args) {
           Scanner sc=new Scanner(System.in);
           String s=sc.nextLine();
           String[] arr=s.split(" ");
                List<Integer> ls=new ArrayList<Integer>();
           for(int i=0;i<arr.length;i++){
              int x=Integer.parseInt(arr[i]);
              ls.add(x);
           }
                System.out.println(ls);
                Collections.sort(ls);
                System.out.println(ls);


        }
}
```

6. Create a method that returns collection that contain only unique String object in the sorted order.

Class Name	UniqueCollection
Method Name	getCollection
Method Description	Accepts a String array and load the elements into a collection that can hold only unique element in a sorted order.
Argument	String []elements
Return Type	Interface type of the Collection used
Logic	Accept a String array, convert it to a collection of unique elements stored in sorted order and return the results.

Solution=>

```
import java.util.*;
```

```
public class Main
{
	public static void main(String[] args) {
		List<Integer> ls=new ArrayList<Integer>();
		ls.add(11);
		ls.add(2);
		ls.add(2);
		ls.add(1);
		ls.add(3);
		ls.add(4);
		System.out.println(ls);
		Set<Integer> s=new TreeSet<Integer>(ls);
		System.out.println(s);

	}
}
```

7. Create a class which accepts a HashMap and returns the keys in the Map

Class Name	MapManager
Method Name	getKeys
Method Description	Returns the keys in the hasp map
Argument	HashMap
Return Type	Set
Logic	Retrieve the keys in hash map and return the set of keys

Solution=>

```
import java.util.*;
public class MapManager
{
	public static void main(String[] args) {
		Map<Integer,String> h=new HashMap<Integer,String>();
		h.put(1,"Dog");
		h.put(2,"Cat");
		h.put(3,"Monkey");
		h.put(4,"Dog");
		System.out.println(h);
		Set<Integer> keys = h.keySet();
		for ( Integer key : keys ) {
		System.out.print( key+" " );
		}

	}
}
```

8.

9. Create a method that returns the current date in the format specified

Class Name	DataGenerator
Method Name	getDate
Method Description	Returns the current date
Argument	String format
Return Type	String date
Logic	Return the current date in the specified format

Solution-

```
import java.time.format.DateTimeFormatter;
import java.time.LocalDateTime;
public class Main {

  public static void main(String args[]) {
    DataGenerator d=new DataGenerator();
    d.getDate();
  }
}
class DataGenerator{
   public void getDate(){
   DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");
   LocalDateTime now = LocalDateTime.now();
   System.out.println(dtf.format(now));

   }
}
```

10. Create a method that calculates the age of a person based on his date of birth

Class Name	AgeCalculator
Method Name	calculateAge
Method Description	Returns the age of the person
Argument	String dob,String format
Return Type	int age
Logic	Returns the age of the person based on his date of birth

Solution=>

```
import java.io.*;
import java.util.*;
```

```
class AgeCalculator {
        static void calculateAge (int current_date, int current_month,
                                        int current_year, int birth_date,
                                        int birth_month, int birth_year)
        {
                int month[] = { 31, 28, 31, 30, 31, 30, 31,
                                                        31, 30, 31, 30, 31 };
                if (birth_date > current_date) {
                        current_month = current_month - 1;
                        current_date = current_date + month[birth_month - 1];
                }
                if (birth_month > current_month) {
                        current_year = current_year - 1;
                        current_month = current_month + 12;
                }
                int calculated_date = current_date - birth_date;
                int calculated_month = current_month - birth_month;
                int calculated_year = current_year - birth_year;

                // print the present age
                System.out.println("Present Age");
                System.out.println("Years: " + calculated_year +
                        " Months: " + calculated_month + " Days: " +
                        calculated_date);
        }
        public static void main(String[] args)
        {
           Scanner sc=new Scanner(System.in);
                // present date
                System.out.println("Enter the Present Age in format dd// mm// yyyy");
                int current_date = sc.nextInt();
                int current_month = sc.nextInt();
                int current_year = sc.nextInt();

                // birth dd// mm// yyyy
                System.out.println("enter the date of birth in format dd// mm// yyyy");
                int birth_date = sc.nextInt();
                int birth_month = sc.nextInt();
                int birth_year = sc.nextInt();

                // function call to print age
                calculateAge (current_date, current_month, current_year,
                        birth_date, birth_month, birth_year);
        }
}
```