Class Jar<T>

java.lang.Object
Jar<T>

Type Parameters:

`T` - the type parameter

```
public class Jar<T>
extends java.lang.Object
```

The type Jar.

Field Summary

Fields

Modifier and Type	Field	Description
`private int`	**`count`**	
`private T[]`	**`list`**	

Constructor Summary

Constructors

Constructor	Description
`Jar()`	Instantiates a new Jar.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
`void`	**`add(T items)`**	Add.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

list

```
private T[] list
```

count

```
private int count
```

Constructor Detail

Jar

```
public Jar()
```

Instantiates a new Jar.

Method Detail

add

isEmpty

```
public boolean isEmpty()
```

Is empty boolean.

Returns:

the boolean

drawItem

```
public T drawItem()
```

Draw item t.

Returns:

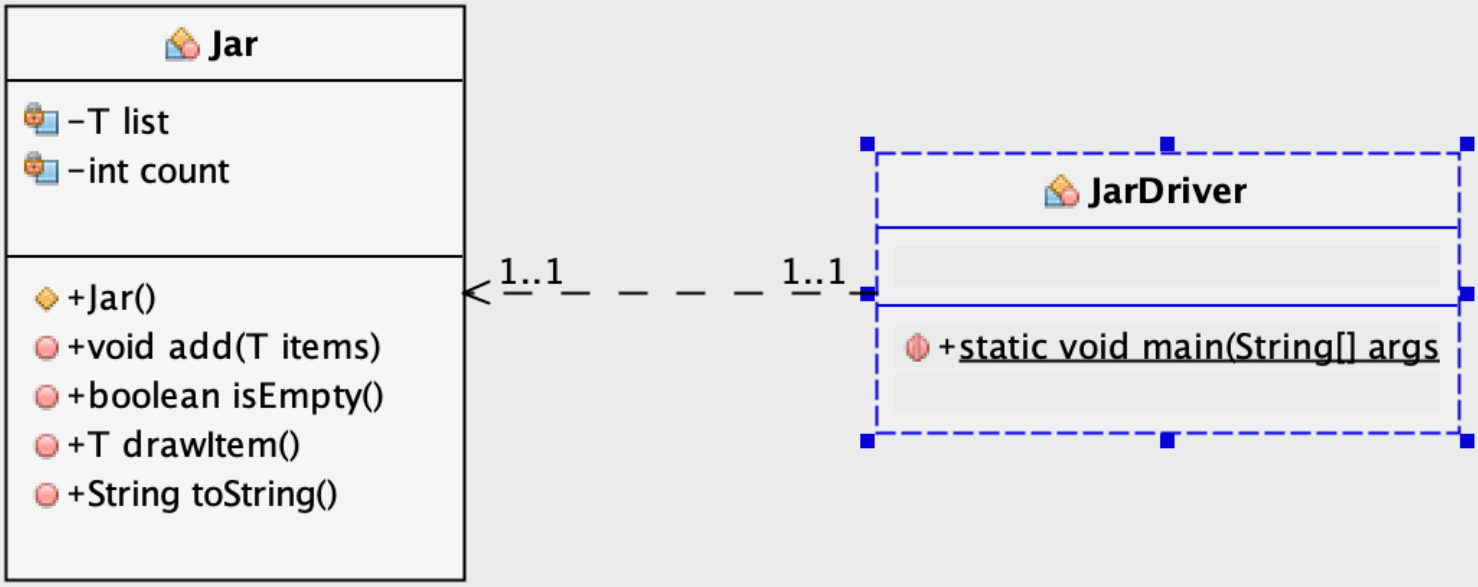
the t

toString

```
public java.lang.String toString()
```

Overrides:

`toString` in class `java.lang.Object`


Jar
-T list
-int count
+Jar()
+void add(T items)
+boolean isEmpty()
+T drawItem()
+String toString()
1..1
1..1
JarDriver
+static void main(String[] args

THOMPSON RIVERS UNIVERSITY

COMP 1230 – Programming 2

Assignment #8 Generics

Vu Nguyen
T00612390
Nov, 19

Problem:

The purpose of this problem is to practice using a generic Jar class.

Create a generic class, called Jar, with a type parameter that simulates drawing an item at random out of a Jar. For example the Jar might contain Strings representing names written on a slip of paper, or the Jar might contain integers representing a random drawing for a lottery. Include the following methods in your generic class, along with any other methods you'd like:

- an add( ) method that allows the user to add **one** object of the specified type
- an isEmpty( ) method (returns true if the Jar is empty, otherwise returns false)
- a drawItem( ) method that randomly selects an object from the Jar and returns it. Return null if the Jar is empty. Do not delete the selected item. (see Note below)
- a toString( ) method (returns a String containing the Jar's contents)

Requirements:

Your Jar class with need an array of size 10 to hold the objects in the Jar, and a count variable to maintain a count of how many objects are actually in the Jar.

In the driver file that tests your class create 2 Jars, one with the names of 6 of your friends, the other with numbers between 2 and 8 inclusive representing the number of hours you will spend partying tonight with 3 of your friends.

Use the add( ) method to populate the 2 Jars, and the drawItem( ) method for each Jar to determine i) which 3 friends you will invite out to party with and ii) how many hours of partying you and your friends will do.

Test Cases:

Test all method- add, drawItem, toString, isEmpty
Test CTor and make sure its doing what a Generic type should do
Test for random in names and hours drawing

```java
/*
* Student name: Vu Nguyen
* Student Number: T00612390
* Due Date: Nov 18
* Program Description: This program is to simulate the action of drawing out items from a hat
*                      ie: in this case its being used to create a "jar" filled with names and a " jar"
*                      filled with durations of time of which you will spend with these names.
*
*/
import java.util.Random;
import java.util.Arrays;

/**
 * The type Jar.
 *
 * @param <T> the type parameter
 */
public class Jar<T>
   {

   private T list[];
   private int count;

   /**
    * Instantiates a new Jar.
    */
//creates 10 empty Jar spaces
   public Jar()
   {
      list = (T[])new Object[10];
      count=0;

   }

   /**
    * Add.
    *
    * @param items the items
    */
//add–adds items to list if space permits
   public void add(T items)
   {
      //checks if list has space to add more
      if (list.length > count)
         {

         list[count] = items;
         count++;

         }
```

```
51      }
52
53      /**
54       * Is empty boolean.
55       *
56       * @return the boolean
57       */
58  //isEmpty checks if the array is empty
59      public boolean isEmpty()
60      {
61          return (count == 0);
62
63      }
64
65      /**
66       * Draw item t.
67       *
68       * @return the t
69       */
70      public T drawItem()
71      {
72          //checks if array is populated or not
73          if (isEmpty())
74              {
75              return null;
76              }
77          Random random = new Random();
78          int i = random.nextInt(count);
79          //returns every item in list
80          return list[i];
81
82
83      }
84
85      @Override
86      public String toString()
87      {
88          T items[] = (T[]) new Object[count];
89
90          for (int i = 0; i < count; i++)
91
92              items[i] = list[i];
93              return Arrays.toString(items);
94
95      }
96
97
98
99      }
```

```
1
2  /*
3   * Student name: Vu Nguyen
4   * Student Number: T00612390
5   * Due Date: Nov 18
6   * Program Description: This file is the driver program for Jar
7   *
8   */
9  public class JarDriver
10    {
11     /**
12      * The entry point of application.
13      *
14      * @param args the input arguments
15      */
16     public static void main(String[] args)
17     {
18        //creates Jar for friends and hour
19        Jar<String> friendsJar = new Jar<String>();
20
21        Jar<Integer> hoursJar = new Jar<Integer>();
22
23        //sees if isEmpty is working for both Jars
24        System.out.println("Checking if friendsJar is empty: " + friendsJar.isEmpty());
25        System.out.println("Checking if hoursJar is empty: " + hoursJar.isEmpty());
26
27        // add elements to friendJar
28        friendsJar.add("Karen");
29        friendsJar.add("Mike");
30        friendsJar.add("Pike");
31        friendsJar.add("Susan");
32        friendsJar.add("Bob");
33        friendsJar.add("Dell");
34
35        // add elements to hoursJar
36        hoursJar.add(2);
37        hoursJar.add(3);
38        hoursJar.add(4);
39        hoursJar.add(5);
40        hoursJar.add(6);
41        hoursJar.add(7);
42        hoursJar.add(8);
43
44        System.out.println("itms have been added, Checking if hoursJar is empty: " +
   hoursJar.isEmpty());
45
46        // draw 3 items from friendsJar
47        System.out.println("Picking 3 friends from Jar: " + friendsJar.drawItem() +" "+
48                           friendsJar.drawItem()+ " " + friendsJar.drawItem());
49
50        // draw one item from hoursJar
51        System.out.println("Picking time from Jar: "+ hoursJar.drawItem());
```

```
52
53          // display the items in friendsJar, hoursJar
54          System.out.println("\nPrinting friendsjar without toString : \n" +friendsJar);
55          System.out.println("Printing hoursJar without toString : \n" +hoursJar);
56
57          System.out.println("testing toString for friendsJar " + friendsJar.toString());
58          System.out.println("testing toString for hoursJar " + hoursJar.toString());
59      }
60
61
62      }
63
```

```
Checking if friendsJar is empty: true
Checking if hoursJar is empty: true
itms have been added, Checking if hoursJar is empty: false
Picking 3 friends from Jar: Mike Dell Karen
Picking time from Jar: 3

Printing friendsjar without toString :
[Karen, Mike, Pike, Susan, Bob, Dell]
Printing hoursJar without toString :
[2, 3, 4, 5, 6, 7, 8]
testing toString for friendsJar [Karen, Mike, Pike, Susan, Bob, Dell]
testing toString for hoursJar [2, 3, 4, 5, 6, 7, 8]
```

```
Checking if friendsJar is empty: true
Checking if hoursJar is empty: true
itms have been added, Checking if hoursJar is empty: false
Picking 3 friends from Jar: Karen Pike Susan
Picking time from Jar: 6

Printing friendsjar without toString :
[Karen, Mike, Pike, Susan, Bob, Dell]
Printing hoursJar without toString :
[2, 3, 4, 5, 6, 7, 8]
testing toString for friendsJar [Karen, Mike, Pike, Susan, Bob, Dell]
testing toString for hoursJar [2, 3, 4, 5, 6, 7, 8]
```

```
Checking if friendsJar is empty: true
Checking if hoursJar is empty: true
itms have been added, Checking if hoursJar is empty: false
Picking 3 friends from Jar: Bob Karen Mike
Picking time from Jar: 3

Printing friendsjar without toString :
[Karen, Mike, Pike, Susan, Bob, Dell]
Printing hoursJar without toString :
[2, 3, 4, 5, 6, 7, 8]
testing toString for friendsJar [Karen, Mike, Pike, Susan, Bob, Dell]
testing toString for hoursJar [2, 3, 4, 5, 6, 7, 8]
```