

Name \_\_\_\_\_ Role (circle one) programmer/computer/project manager

Name \_\_\_\_\_ Role (circle one) programmer/computer/project manager

Name \_\_\_\_\_ Role (circle one) programmer/computer/project manager

Name \_\_\_\_\_ Role (forth person only) quality control

## Making Dixie Cup Arrays

### Your Tasks (Mark these off as you go)

- ☐ Create a DixieCup class and a DixieCupMaker class
- ☐ Declare variables items, itemNumber
- ☐ Create the DixieCup constructor
- ☐ Add the addItem method to the DixieCup class
- ☐ Add the getItems method to the DixieCup class
- ☐ Add the removeItem method to the DixieCup class
- ☐ Write the main method in the DixieCupMaker class
- ☐ Create an array of DixieCups
- ☐ Add DixieCups to your Array
- ☐ Add items to your Dixie Cups
- ☐ Have Ms. Pluska check off your DixieCup and DixieCupMaker classes before you continue
- ☐ Complete challenges 1 thru 5
- ☐ Have Ms. Pluska check off your challenges 1 thru 5 before you continue
- ☐ Return your Materials to Ms. Pluska
- ☐ Create a project in NetBeans called DixieCupMaker
- ☐ Receive credit for this project

### ☐ Create a DixieCup class and a DixieCupMaker class

Locate the two large sheets of paper at your assigned location

On the first sheet of paper,

- Write "DixieCup class" at the top of the page
- Declare the DixieCup class using the appropriate signatures

On the second sheet of paper,

- Write "DixieCupMaker class" at the top of the page
- Declare the DixieCupMaker class using the appropriate signatures.

Your papers should like the example below,

Sheet 1	Sheet 2
<u>DixieCup class</u> public DixieCup{  //Leave lots of space here  }	<u>DixieCupMaker class</u> public DixieCupMaker{  //Leave lots of space here  }

## □ Declare variables items, itemNumber

At the top of the DixieCup class we will declare a private variable called items. This variable will be an array that stores the names of the items in the cup. To do this write the following code at the top of the DixieCup class,

```
private String items[];
```

As we add items to the cup, we will need to keep track of the items and where they are stored in the items array. We will do this with an itemNumber variable. For now just assign the value of itemNumber to 0 as shown below.

```
private int itemNumber = 0;
```

## □ Create the DixieCup constructor

Before we can start creating Dixie Cups we need a constructor. The purpose of the DixieCup class is to create Dixie Cups of random contents. To do this we need to keep track of the number of items the cup can hold. Once we get the number of items we can initialize the items array we declared earlier.

```
public DixieCup (int i){  
    items = new String[i];  
}
```

Each portion of the constructor is defined below,

**public** - refers to the accessibility of the constructor. Because the constructor is declared as public it can be accessed by other classes.

**DixieCup** - refers to the name of the constructor. It must be identical to the name of the class

**int i** - refers to an int type parameter called "i". When the constructor is implemented a value for "i" must be provided.

**items = new String[i]** - Initializes the items array to the number of items that will be stored in the cup

Write the DixieCup constructor in your DixieCup class.

## □ Add the addItem method to the DixieCup class

Now that we have a constructor, we can start adding contents to our Dixie cups.

We will do this by adding a method called addItem. The job of addItem is to simply add items to the cup. It does not need to return any information about the item, so this method will be a "void" type method as shown below.

```
public void addItem(String n){  
    items[itemNumber] = n;  
    itemNumber++;  
}
```

Each portion of the addItem method is defined below,

`public` - refers to the accessibility of the method. Because the method is declared as `public` it can be accessed by other classes.

`void` - indicates that the job of this method is "to do" something, it does not return any values

`addItem` - refers to the name of the method

`String n` - refers to an `String` type parameter called "n". When `addItem` is implemented a value for "n" must be provided.

`items[itemNumber] = n` - sets the name of the item in the `items` array at the position of the `itemNumber`.

`ItemNumber++` - increments the `itemNumber` each time `addItem` is called.

Write the `addItem` method in your `DixieCup` class

## □ Add the `getItem` method to the `DixieCup` class

Now that we can add items, it would be nice to know the contents of our cups. To see the items in our cup we will create a `getItem` method. The job of the `getItem` method is simply to *return* the item at a specified index. Because the job of this method is to return something we need to specify that in the signature as shown below,

```
public String getItem(int i){  
    return items[i];  
}
```

Each portion of the `getItem` method is defined below,

`public` - refers to the accessibility of the method. Because the constructor is declared as `public` it can be accessed by other classes.

`String` - refers to the what the method is returning. Whatever is returned can be accessed by other parts of the program. In our case, it is a `String` at specified index in the `items` array.

`getItem()` - refers to the name of the method

`int i` - refers to the index of the item to be returned

`return items[i]` - returns the item at the specified index

Write the `getItem` method in your `DixieCup` class

## □ Add the `removeItem` method to the `DixieCup` class

Now that we know how to add items to the cups and see the items. We want to have a method to remove them. Again, the job of `removeItem` is simply to remove an item, it does not return anything.

The signature for this method will look as follows,

```
public void removeItem(String n)
```

But, how do we know where this item is in the `items` array? To figure this out, we will need to loop over the `items` array until we find a match between the item specified by the user and the item in the array. Then once we find the item, we will set its value equal to "null". An example of how to do this is shown below,

```
for(int i = 0; i < items.length;i++){  
    if(items[i].equals(n)){  
        items[i] = null;  
    }  
}
```

Write the `removeItem` method in your `DixieCup` class.

## □ Write a main method in the DixieCupMaker class

Obtain a tray of supplies from Ms. Pluska. In your tray you will notice Dixie cups and some random items.

You can now use your DixieCup class to create Dixie cups full of items!

Locate your DixieCupMaker class and write a main method like shown below,

```
public static void main(String args[]){  
    //leave lots of space here!  
}
```

## □ Create an array of DixieCups

The array of DixieCups you create can only contain the number of DixieCups you have been provided. For example, you have been provided 5 Dixie cups, you array can only be 5 items long. Declare an array of Dixie Cups in the main method of your DixieCupMaker class like that shown below. Be sure to substitute the number 5 with the number of cups you have been provided.

```
DixieCup[] cupsArray = new DixieCup[5];
```

## □ Add Dixie cups to your array

The array you just declared has empty spots reserved for you Dixie cups. To keep track of your Dixie cups use the key attached to this activity.

You will notice on the key that there are circles for each Dixie cup (you may have more circles than cups). As you add Dixie Cups to your array, place them on the appropriate circle.

To add a Dixie Cup to your array, simply specify the location in the array, then create a new DixieCup object at that location. In the example below, a new DixieCup object is created at position 2. The DixieCup object can hold 5 things.

```
cupsArray[2] = new DixieCup(5);
```

Using the example above, my key looks as follows,



Follow the example above to write code for adding Dixie Cups to the CupsArray you created. Then place the cups on the appropriate circles on the key.


## □ Add items to your Dixie Cups

To add items to your cups, you will use the addItem method. For example, to add items to the Dixie Cup I created above I could write,

```
cupsArray[2].addItem("paper clip");
```

As you add items, be sure to keep track of the items and the order in which you add them.

Using the example above, my key looks as follows

	0	1		3	4
position	items	items	items	items	items
0			Paper clip		
1					
2					

Follow the example above to add items to your Dixie Cups. Remember to both write the code and keep track of the items added on the key provided.

## □ Remove items from your Dixie Cups

To remove items from your cups, you will use the `removeItem` method. For example, to remove an item from the Dixie Cup I created above I could write,

```
cupsArray[2].removeItem("paper clip");
```

As you remove items, be sure to keep track of the items.

Using the example above, my key looks as follows

	0	1		3	4
position	items	items	items	items	items
0					
1					
2					

Follow the example above to remove items from your Dixie Cups. Remember to both write the code and keep track of the items removed on the key provided.

## □ Have Ms. Pluska check off your DixieCup and DixieCupMaker classes before you continue



Before you continue have Ms. Pluska check off your DixieCup and DixieCupMaker classes

## □ Complete Challenges 1 thru 6

### Challenge 1

There are two problems with the `addItem` method. First, if you add items beyond the length of the declared items array you will go out of bounds. Second, if you remove an item, how might you add an item to the "empty" location? Rewrite this method to account for this problem.

### Challenge 2

In the DixieCup class write a method called toString() which returns the contents of the Dixie cup as a String. In the DixieCupMaker class write code that will print the contents of each DixieCup object in your cupsArray.

### Challenge 3

In the DixieCup class, write a method called setItem() that sets an item in a cup to a different item. For example, what if I wanted to change the paper clip to a toothpick? This method should accept an int variable as a parameter. This variable corresponds to the index of the item to be set. Call this method in the DixieCupMaker class and keep track of your Dixie cup items on the key provided.

### Challenge 4

In the DixieCup class, write a method called swapItems() that swaps an item in one cup with another item from a different cup. The method should accept three parameters. The first parameter is the DixieCup object you want to swap with. The next parameters are integers which represent the indices of the items to be swapped. Call this method in the DixieCupMaker class and keep track of your Dixie cup items on the key provided.

### Challenge 5

In the DixieCup class, write a method called numItems() that returns the total number of items in the cup. Your method should ignore items with null values.

### Challenge 6

Locate the DixieCup with the most items, then print the location of the cup  
Locate the DixieCup with the least items, then print the location of the cup

## ☐ **Have Ms. Pluska check off challenges 1 thru 6 before you continue**



Before you continue have Ms. Pluska check off challenges 1 thru 6. \_\_\_\_\_

## ☐ **Return your materials to Ms. Pluska**

Return your tray of materials to Ms. Pluska to the designated location

## ☐ **Create a project in NetBeans called DixieCupMaker**

Create a project in NetBeans called DixieCupMaker. Locate on the DixieCupMaker project description on the course website and complete the project requirements.

## ☐ **Receive Credit for this Project**

Submit this project guide to the needs to be graded folder to receive credit for the group portion of this project.

Submit your NetBeans project on Zofia to receive credit for the individual portion of this project.

