

## CS125 Fall 2009 Exam 3 Practice Exam

Before you start, please note: it is possible to get strong clues for answering some of the questions in the review by referring to other questions. For example, converting from code to a flow chart and vice versa. In general, you should not depend on both versions of a question showing up in the exam.

Also note: this study guide is not comprehensive. I strongly suggest reviewing: the code examples composed in class, homework assignments, material from the last exam & the self check questions in the text.

1. Write & document a public method called **reverseArray** that takes an array of Strings as a parameter and reverses the contents such that the first item becomes the last item.

```
// code goes here
```

2. Write & document a public method called **getBiggest** that takes an **ArrayList** of **Doubles** as a parameter and find the largest value. If the ArrayList is empty, print some sort of error message & return zero.

```
// code goes here
```

3. Examine the code below & explain the difference between **copy1** & **copy2**. Provide an alternative that achieves the same result as the third line (that is, provide code that creates a copy which would have the same properties as **copy2**).

```
int[] data = new int[] {5, 10, 15, 20};
```

```
int[] copy1 = data;
```

```
int[] copy2 = data.clone();
```

4. Examine the code below. Add the necessary lines of code such that each new Player object will get its own unique idNum. Note: this should not be done by changing any of the existing code.

```
public class Player
{
    // instance fields
    int score;
    int idNum;

    /**
     * constructor - sets idNum
     */
    public Player()
    {

    }

    /**
     * returns idNum
     *@return the objects idNum
     */
    public int getIdNum()
    {
        return idNum;
    }

}
```

5. Write a **for each** loop that sums up the contents for an **ArrayList** called **values**, stores the result in double called **sum** & then prints it.

```
// code goes here
```

6. Write a **for** loop that finds the smallest value in an array called **values**, and swaps it to the last position of the array

```
// code goes here
```

7. Write a **while** loop that examines all the values in an integer array called **values**, finds the count of the values that are even & then prints it.

```
// code goes here
```

8. Define encapsulation.

9. Define cohesion.

10. What is a postcondition? Give an example.

11. What does shadowing mean? Give an example.

12. Write & document a class called **ArrayChecker** with a static method called **mostFrequentItem**. It should take an array of integers & return the item that occurs the most often. If more than one item is 'tied for first place,' return the one that is encountered first in the array. Including a precondition comment indicating the method does not take empty or null arrays. Note: this problem may be easier if you split up the work into to more than one method. You might also want to check our dice game code.

Examples:

the mostFrequentItem of {1,2,3,1,4,5,1,6,7} would be 1

the mostFrequentItem of {1,3,2,4,4,3,4,6,3} would be 3