Name: _____

CS 141: Introduction to (Java) Programming: Exam 2 - Solutions

Jenny Orr • Willamette University • Fall 2015

1.	(max 16)	3.	(max 30)
2.	(max 30)	4.	(max 24)
Total:			(max 100)

1. (16 pts total) Given the code below, complete the output as started for you on the right:

```
public class MethodTesting {
    public static void main(String[] args) {
       int [] a = {3,4,5};
       System.out.println("a[0]="+ a[0]);
       arrayChange(a);
       System.out.println("a[0]="+ a[0]);

       int s = 4;
       System.out.println("s = " + s);
       intChange(s);
       System.out.println("s = " + s);

       Card card1 = new Card(0);  // Ace of hearts
       System.out.println("card1 = " + card1);
       objectChange1(card1);
       System.out.println("card1 = " + card1);

       Card card2 = new Card(0);  // Ace of hearts
       System.out.println("card2 = " + card2);
       objectChange2(card2);
       System.out.println("card2 = " + card2);
    }
    public static void arrayChange(int [] a) {
        a[0] = a[0]*2;
    }
    public static void intChange(int x) {
        x = x * 2;
    }
    public static void objectChange1(Card c) {
        c.setIndex(1);  // 2 of hearts
    }
    public static void objectChange2(Card c) {
        c = null;
    }
}
```

Output:

```
a[0]= 3

a[0]= 6

s = 4

s = 4

card1 = A♥

card1 = 2♥


card2 = A♥

card2 = A♥
```

2. (1 pts each, 30 pts total) True and False: Please circle **T or F**
(Credit is only given if the instructor can clearly tell which answer is circled)

1) T **or** **F:** The keyword word `static` is used to indicate instance methods and variables.

2) **T** **or** F: A class cannot have more than one superclass.

3) **T** **or** F: In an abstract class, not all of the methods are implemented.

4) **T** **or** F**:** If one changes the value of a class variable, the value is changed for all objects of that type.

5) T **or** **F:** If CEO is a subclass of Employee, then it is ok to have the declaration:

```
CEO e = new Employee();
```

6) T **or** **F:** An object may be created from an interface.

7) T **or** **F:** The keyword `super` is used to call the constructor of a subclass.

8) T **or** **F**: To create a subclass, one uses the `implements` keyword.

9) T **or** **F**: A subclass has access to private instance variables of its superclass.

10) **T** **or** F: A class can implement more than one interface.

11) T **or** **F:** Object parameters are passed by value.

12) **T** **or** F: A subclass inherits data and behavior from a superclass.

13) T **or** **F:** Once an object is garbage collected, it can still be retrieved if needed again.

14) **T** **or** F**:** Once an array is created, its size cannot be changed.

15) T **or** **F:** Once an ArrayList is created, its size cannot be changed.

16) T **or** **F:** When an array is a parameter for a method, both its type and size must be specified in the method header.

17) **T** **or** F**:** Arrays are passed by reference to a method.

18) T **or** **F:** A protected member of a child class may be directly accessed by its parent class.

19) **T** **or** F**:** All methods specified by an interface should be public.

20) **T** **or** F**:** If `Course` is an interface, the following code is allowed:

```
Course[] c = new Course[10];
```

21) **T or** F: If a subclass constructor does not explicitly call a superclass constructor, Java will automatically call the superclass's default constructor.

22) **T or** F: In Java, a reference variable is polymorphic because it can reference objects of types different from its own, as long as those types are related to its type through inheritance.

23) **T or** F: When a class contains an abstract method, you cannot create an instance of the class.

24) **T or** F: An `actionPerformed` method is called when a button-click event is generated.

25) **T or** F: The key word `this` is the name of a reference variable that an object can use to refer to itself.

26) T **or F:** A class's static methods can access instance member variables.

27) **T or** F: An object of a class does not have to exist in order to execute static methods of the class.

28) **T or** F: An exception is an object that is generated as the result of an error or an unexpected event.

29) T **or F:** The `try` clause follows the `catch` clause.

30) **T or** F: An object with no references pointing to it is garbage collected.

3. (6 pts each, 30 pts total) Arrays of primitives and objects:
   a. Write code that declares and creates a 1D array of 100 Die *objects* (remember the 3-step process)? Call the array `dice`. (Assume the Die class has a no-arg constructor that sets the number of sides to 6, and randomly rolls the die.)

```
Die [] dice = new Die[100];
for (int i=0; i < dice.length; i++) {
    dice[i] = new Die();
}
```

   b. Write code that swaps the Die object at array index 0 with the Die object at index 1 in the above `dice` array.

```
Die temp = dice[0];
dice[0] = dice[1];
dice[1] = temp;
```

c. Write code that creates a 2D array of *doubles* called nums which has 3 rows, where the 0th row contains n0 elements, the next row contains n1 elements, and the next row contains n2 elements.

```
double[][] nums = new double[3][];
nums[0] = new double[n0];
nums[1] = new double[n1];
nums[2] = new double[n2]
```

d. Write code that uses ***a nested loop*** to set the values in nums to random numbers.

```
for (int row = 0; row < nums.length; row++) {
    for (int col = 0; col < nums[row].length; col++) {
        nums[row][col] = Math.random();
    }
}
```

e. Write code that would swap *row* 0 with *row* 1 in nums. ***This should be done using array references only***.

```
double [] temp = nums[0];
nums[0] = nums[1];
nums[1] = temp;
```

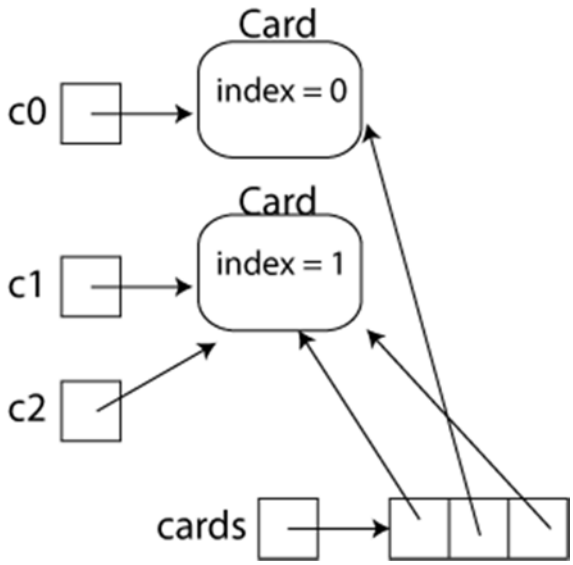
4. (24 pts total) **Object Diagrams:** Assume there exists a Card class containing a single member variable called `index` which stores the card index as was done in the lab. Also assume the following:
   - Hearts is the first suit - the Card's toString will print A♥ for index = 0, 2♥ for index = 1, etc.
   - The Card class contains the normal collection of methods: constructors, setters, getters, toString, copy constructor, etc

   Below, you will be asked to draw the object diagrams. Use the drawing style used in class, e.g. *rectangular boxes indicate object references; use rounded boxes to indicate objects as shown below on the right.*

a) (8 pts) Given the code below:


```
Line 1:     public static void main(String[] args) {
Line 2:       Card c0 = new Card(0); // Ace of hearts, index = 0
Line 3:       Card c1 = new Card(1); // 2 of hearts, index = 1
Line 4:       Card c2 = c1;
Line 5:       Card [] cards = { c1, c0, c2};
Line 6:    }
```

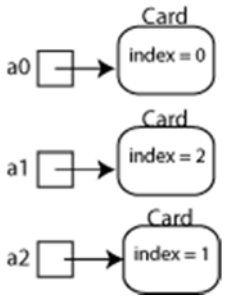
Draw the object diagram for all of objects that exist at Line 6. Include the value of each Card's index.



b) (16 total pts) Given the code below

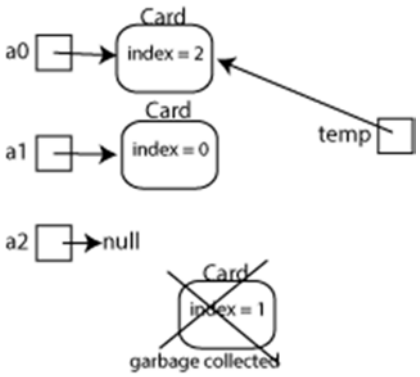
```
Line 1:     public static void main(String[] args) {
Line 2:       Card a0 = new Card(0);
Line 3:       Card a1 = new Card(1);
Line 4:       Card a2 = new Card(a1); // copy constructor
Line 5:       a1.setIndex(2);
Line 6:       System.out.println(a0+","+a1+","+a2);
Line 7:       Card temp = a1;
Line 8:       a1 = a0;
Line 9:       a0 = temp;
Line 10:      a2 = null;
Line 11:      System.out.println(a0+","+a1+","+a2 + " " + temp);
Line 12:    }
```

i. (6 pts) Draw the object diagram as it would appear at Line 6. Include the value of each Card's index.

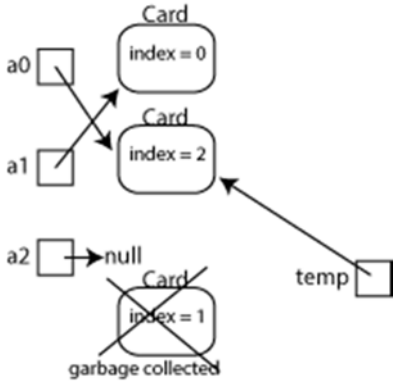


ii. (2 pts) What is printed at Line 6? A♥ 3♥ 2♥

iii. (6 pts) Draw the object diagram as it would appear at Line 11. (It may help to use the scratch area to work out the details and then recopy neatly to the left). Indicate what, if anything, is garbage collected.



Scratch area:



iv. (2 pts) What is printed at Line 11? 3♥ A♥ null 3♥