Name: _____

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

Jenny Orr • Willamette University • Spring 2018

1.	(max 14)	3.	(max 21)	5.	(max 9)	7.	(max 16)
2.	(max 10)	4.	(max 14)	6.	(max 16)		
Total:							(max 100)

1. (1 pts each, 14 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** In an abstract class, not all of the methods are implemented.

   2) **T or F:** If Electric is a subclass of Vehicle, then it is ok to have the declaration:

   ```
   Electric e = new Vehicle();
   ```

   3) **T or F:** An object can be created from an abstract class.

   4) **T or F:** A do-while loop always executes the loop body at least once.

   5) **T or F:** Arrays can be used as arguments or return values in methods.

   6) **T or F:** The size of an ArrayList cannot be changed.

   7) **T or F:** When declaring a multi-dimensional array, the size of all of the dimensions must be specified.

   8) **T or F**: The implements keyword is used to specify that a class uses an interface.

   9) **T or F:** Array parameters are passed (i.e. called) by value.

   10) **T or F:** Overloading is when methods in a class have the same name but a different list of parameters.

   11) **T or F:** An example of polymorphism is when an array contains objects of different types but where those types are related through inheritance.

   12) **T or F:** An exception is an object that is thrown when an error or an unexpected event occurs during runtime.

   13) **T or F:** A deep copy of an object creates new references but not necessarily new objects.

   14) **T or F:** An object cannot be created from only an interface.

2. (10 pts) **Looping:** Write a **<u>while-loop</u>** which prints out the even numbers from 200 down to 100. That is, the output should look something like: 200 198 196 194 ... 100.

3. (7 pts each, 21 pts total) **1D Arrays:** Write code that:

   a. Declares and creates a 1D array called `vals` consisting of 100 doubles*:*

b. Implement a **for-loop** that initializes each array element in `vals` to a random number (e.g. use Math.random()) .

c. Implement a second for-loop which computes and prints the maximum value of all of the elements in `vals`.

4. (14 pts total) **2D Arrays**:
    a. (6 pts) Write code that declares and creates a 2D array of Card objects called `myCards` with 10 rows and 20 columns.

    b. (8 pts) Write a nested loop to initialize all the elements of the array. Do not use the magic numbers 10 or 20, instead make use of the length variable.

5. (9 pts) **Class vs Instance variables:** Given the class:

```
public class Bag {
    private static int size = 10;
    private String item;

    public Bag(String item) {
          this.item = item;
    }

    public static void setSize(int s) {
        size = s;
    }

    public void setItem(String item) {
        this.item = item;
    }

    public String toString() {
        return item + ", size = " + size;
    }
}
```

What is the output of the following code at Lines 0, 1, and 2:

```
public class Exam2Project {
   public static void main(String[] args) {
       Bag chips = new Bag("Chips");
       Bag donuts = new Bag("Donuts");
Line 0:  System.out.println(chips + " and " + donuts);

       chips.setItem("Corn chips");
       chips.setSize(12);

Line 1:  System.out.println(chips + " and " + donuts);
       Bag.setSize(15);
Line 2:  System.out.println(chips + " and " + donuts);
    }
}
```

Output:

Line 0: _____

Line 1: _____

Line 2: _____

6. (16 pts total) **Object Diagrams:** Assume the Card class is the same as the one you wrote in the lab assignment. That is, it contains one member variable: the index of the card. Its toString() method prints out the name of the card, e.g. ace of spades.

   Given the code below:

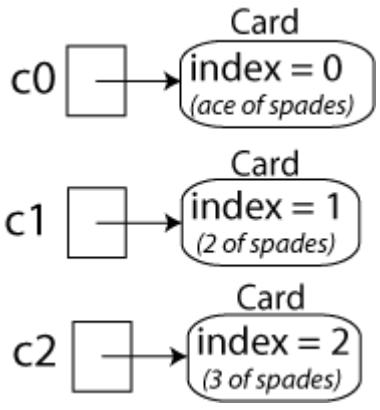
```
Line 0:   public class CardProject {
Line 1:        public static void main(String[] args) {
Line 2:             Card c0 = new Card(0);  // ace of spades
Line 3:             Card c1 = new Card(1);  // 2 of spades
Line 4:             Card c2 = new Card(2);  // 3 of spades

Line 5:             Card[] cards = {c0, c1, c2, c1};
Line 6:             System.out.println(cards[0]+",  "+cards[1] +
                           ",  "+cards[2] +",  "+cards[3]);

Line 7:             cards[0]= c1;
Line 8:             cards[1] = cards[2];
Line 9:             cards[2] = cards[0];
Line 10:            cards[3] = c0;


Line 11:            System.out.println(cards[0]+",  "+cards[1] +
                           ",  "+cards[2] +",  "+cards[3]);
Line 12:       }
Line 13:  }
```

   a) (6 pts) The object diagram for Lines 2-4 is shown below. Add to this diagram, the references and objects after Line 5 (but not Lines 6-11) has been executed.



   b) (2 pts) What is printed at Line 6?

c) (6 pts) Redraw the entire object diagram from part a) but modified to reflect the objects and references as they would appear after executing ***all lines of code***.

d) (2 pts) What is printed out at Line 11?

7. (16 pts total) **Inheritance:** Consider the classes shown here:

```
public class Car {

    public void m1() {
      System.out.println("car in m1");
   }

   public void m2() {
      System.out.println("car in m2");
   }

   public String toString() {
      return "vroom ";
   }
}
```

```
public class Truck extends Car {
    public void m1() {
      System.out.println("truck in m1");
   }
   public void m2() {
      super.m1();
   }
   public void m3() {
       System.out.print("truck in m3: ");
       m1();
   }
   public String toString() {
      return "truck: " + super.toString();
   }
}
```

Indicate what is the printed at each numbered line below, *or indicate if the line results in an error*.

```
                public class InheritanceProject {
                    public static void main(String[] args) {
                        Car car = new Car();
        Line 1:         System.out.println(car);
        Line 2:         car.m1();
        Line 3:         car.m2();
        Line 4:         car.m3();

                        Truck truck = new Truck();
        Line 5:         System.out.println(truck);
        Line 6:         truck.m1();
        Line 7:         truck.m2();
        Line 8:         truck.m3();
                    }
                }
```

Output:

Line 1: _____

Line 2: _____

Line 3: _____

Line 4: _____

Line 5: _____

Line 6: _____

Line 7: _____

Line 8: _____