Name:

Computer Science 1

Final Exam Answer Sheet

Fall 2017

1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

11.

12.

13.

14.

15.

Computer Science 1

Final Exam—Fall 2017

Doodle here.

This is a closed-book, no-calculator, no-electronic-devices, individual-effort exam. You may reference one page of handwritten notes. All answers should be clearly written. Questions that require code should be written using correct Java syntax. You may write `SOP` to represent `System.out.println`.

Class	Method/Constructor	Description
`Scanner`	`Scanner(System.in)`	create `Scanner` for parsing `System.in`
	`Scanner(String text)`	create `Scanner` for parsing `text`
	`String next()`	get next delimited word
	`double nextDouble()`	get next delimited `double`
	`boolean nextBoolean()`	get next delimited `boolean`
	`int nextInt()`	get next delimited integer
`String`	`int length()`	get number of characters
	`char charAt(int i)`	get the character at index `i`
	`String toUpperCase()`	get a `String` like this one, but in all-caps
	`int indexOf(char c)`	find the index of the first `c`
	`String substring(int a)`	get substring from index `a` to `String`'s end
	`String substring(int a, int b)`	get substring from index `a` to before index `b`
`Math`	`int max(int a, int b)`	get the maximum of `a` and `b`.
	`int min(int a, int b)`	get the minimum of `a` and `b`.
	`double pow(double base, double exponent)`	raise `base` to the `exponent` power.
`Random`	`Random()`	create a random number generator.
	`nextInt(int i)`	get random number in $[0, i-1]$.
	`nextDouble()`	get random number in $[0.0, 1.0)$.
`File`	`File(String path)`	create a `File` object for the given path
	`boolean exists()`	returns true if path refers to an existing file
`ArrayList<T>`	`ArrayList<T>()`	create empty list for holding `T` references
	`boolean add(T e)`	adds `e` to end of list; always returns true
	`void set(int i, T e)`	replaces element `i` with `e`
	`void clear()`	removes all elements
	`T remove(int i)`	removes and returns element `i`
	`int indexOf(T e)`	finds index of first element matching `e`
	`T get(int i)`	gets the item at index `i`
	`int size()`	gets the number of items in the list

1. How many constructors may be defined in a class?

2. Consider the following code:

```
public class Circle {
  private double x;
  private double y;
  private double r;

  public Circle(double givenX, double givenY, double givenRadius) {
    x = givenX;
    y = givenY;
    r = givenRadius;
  }

  public double getDiameter() {
    double diameter = 2 * r;
    return diameter;
  }

  public void move(double dx, double dy) {
    x += dx;
    y += dy;
  }

  public double getArea() {
    double squared = r * r;
    return 3.141592653589793 * squared;
  }
}
```

   Which variables are local variables? You may not need all the blanks.

3. In the preceding code, which variables are instance variables? You may not need all the blanks.

4. In the preceding code, which variables are parameters? You may not need all the blanks.

5. Write the signature of a constructor for class `Donut` that lets programmers indicate whether or not the donut has sprinkles, whether or not the donut has frosting, and who is buying the donut.

6. You see the token `Gichayeog` in some Java source code that follows standard naming conventions.[1] What is it?

7. A method that gets all of its data from parameters or input—and not instance variables—should be marked with what keyword?

8. Complete the blank line in the code below so that the constructor persists the parameter to the instance variable:

```
public class VirtuaPet {

  private String name;

  public VirtuaPet(String name) {
    ___________________
  }
}
```

[1] 기차역 (gichayeog) is the Korean word for *train station*.

9. What is the output of the following code?

```
public static void foo(int i) {
  if (i < 0) {
    throw new IllegalArgumentException(i + " -> :(");
  }
  System.out.println(i);
}

public static void main(String[] args) {
  try {
    foo(0);
    foo(2);
    foo(-3);
    foo(5);
  } catch (IllegalArgumentException e) {
    System.out.println(e.getMessage());
  }
}
```

10. What are the contents of the array `a` after the following code is executed?

```
int[] a = new int[6];
for (int i = 0; i < a.length; ++i) {
  a[i] = 2 * i + 1;
}
```

11. What are the contents of the array `a` after the following code is executed? Mark `null` elements with an X.

```
String s = "hither";
String[] a = new String[8];
for (int i = 0; i < s.length(); ++i) {
  a[i] = s.substring(0, s.length() - i);
}
```

12. What are the contents of the array `a` after the following code is executed?

```
int[] a = {7, 999, 5, 6, 1, 4, 1, 2};
int i = 0;
while (a[i] != 999) {
  int iNext = a[i];
  a[i] = 0;
  i = iNext;
}
```

13. Write class `Quadratic` that models a quadratic polynomial of the form $ax^2 + bx + c$. Give it the following:

    (a) A constructor that accepts three `double`s representing the three coefficients.

    (b) A method `evaluate` that accepts a `double` parameter for a given x value. It evaluates the polynomial at x and returns the result as a `double`.

    (c) A method `toString` that returns a `String` representation of this polynomial, of the form `A * x^2 + B * x + C`, but in which `A`, `B`, and `C` have been replaced by the quadratic's actual values.

Consider this example usage:

```
Quadratic q = new Quadratic(1, 2, 3);
System.out.println(q.evaluate(0)); // prints 3
System.out.println(q.evaluate(1)); // prints 6
System.out.println(q); // prints "1 * x^2 + 2 * x + 3"
```

14. Write class `Boundser` that tracks the minimum and maximum integers that have been added to it. Give it the following:

    (a) A constructor that accepts no parameters.

    (b) A method `add` that accepts an `int` parameter for the value to be added. It expands the minimum and maximum bounds as necessary to envelop this new value.

    (c) A method `getMin` that returns as an `int` the least value that has been added so far. If no values have been added, it returns 0.

    (d) A method `getMax` that returns as an `int` the greatest value that has been added so far. If no values have been added, it returns 0.

    Consider this example usage:

```
Boundser b = new Boundser();
System.out.println(b.getMin()); // prints 0
System.out.println(b.getMax()); // prints 0
b.add(3);
System.out.println(b.getMin()); // prints 0
System.out.println(b.getMax()); // prints 3
b.add(-10);
System.out.println(b.getMin()); // prints -10
System.out.println(b.getMax()); // prints 3
```

15. Write class `RandomCycler` that manages a collection of names. It allows a name to be randomly drawn from the collection. Unlike a raffle, the name is not permanently removed from the collection, but that same name won't be drawn again until all others have been. For example, suppose the collection contains the three names A, B, and C. We might see the following random sequence for 18 calls to `draw`: ABC BAC CBA BCA CAB BAC.

    (a) A constructor that accepts no parameters.

    (b) A method `add` that accepts a `String` parameter for the name to be added. It adds the name such that it will still be drawn within this cycle.

    (c) A method `draw` that returns as a `String` a *random* name that has not yet been drawn in this cycle. If all names have already been drawn in this cycle, it begins a new cycle, in which any name is eligible to be drawn. Additionally, it ensures that this name will not be drawn again until this cycle is complete.

    Consider this example usage:

```
RandomCycler rc = new RandomCycler();
rc.add("Mondovi");
rc.add("Eleva");
System.out.println(rc.draw()); // prints one of Mondovi and Eleva; let's say Eleva
System.out.println(rc.draw()); // prints Mondovi, since Eleva's already been drawn
rc.add("Strum");
System.out.println(rc.draw()); // prints Strum, since Eleva and Mondovi have been drawn
System.out.println(rc.draw()); // new cycle; prints one of Mondovi, Eleva, and Strum
```