Name: ______________________________

Computer Science 145

Final Exam Answer Sheet

Fall 2016

1. a b c d

2. a b c d

3. a b c d e

4. a b c d

5. a b c d

6. ______________________________

7. ______________________________

8. ______________________________

9. a b c d

10. a b c d

11. ______________________________

12. ______________________________

13. ______________________________

14. a b c d

15.

16.

17.

18.

19.

20.

21.

Computer Science 145

Final Exam—Fall 2016

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. Which of the following statements are true of instance variables? Check zero or more.

   (a) They must be marked `static`.

   (b) They should be given their initial assignments by the class's constructor.

   (c) They should be marked `private`.

   (d) They can only be accessed by non-`static` methods.

2. Which of the following classes contain valid constructors? Check zero or more.

   (a)

   ```
   class Monster {
     private int hitPoints;
     public Monster(int hitPoints) {
       return new Monster(hitPoints);
     }
   }
   ```

   (b)

   ```
   class State {
     private String name;
     public void State(String name) {
       this.name = name;
     }
   }
   ```

   (c)

   ```
   class Polycephalus {
     private int nHeads;
     public Polycephalus(int nHeads) {
       this.nHeads = nHeads;
     }
   }
   ```

   (d)

   ```
   class Number {
     private double number;
     public Integer(int i) {
       number = i;
     }
   }
   ```

3. Which of the following methods of the class `ArrayList` are getters/accessors? Check zero or more.

   (a) `size`

   (b) `contains`

   (c) `isEmpty`

   (d) `indexOf`

   (e) `remove`

4. Which of the following methods of the class `ArrayList` are setters/mutators? Check zero or more.

   (a) `set`

   (b) `clear`

   (c) `get`

   (d) `add`

5. Which of the following statements are true `ArrayList`? Check zero or more.

   (a) The `length` property of a list's backing array yields the same value as the list's `size` method.

   (b) A list may allocate many arrays over the list's lifetime.

   (c) `ArrayList` is an object.

   (d) The `ArrayList` class encapsulates algorithms for adding and removing elements, operations which are not directly supported by arrays.

6. Provide an example of a multi-word class name that follows Java case conventions.

7. Provide an example of a multi-word method name that follows Java case conventions.

8. Provide an example of a multi-word variable name that follows Java case conventions.

9. Which of the following assignments make the expression `a || !b` evaluate to true? Check zero or more.

   (a) `a = false`, `b = false`

   (b) `a = true`, `b = false`

   (c) `a = false`, `b = true`

   (d) `a = true`, `b = true`

10. Which of the following assignments make the expression `a && b && !a` evaluate to true? Check zero or more.

    (a) `a = false`, `b = false`

    (b) `a = true`, `b = false`

    (c) `a = false`, `b = true`

    (d) `a = true`, `b = true`

11. Write an expression that evaluates to (not prints, not returns) the number of characters of a `String` named `text`.

12. Write an expression that evaluates to (not prints, not returns) the number of elements in array `nums`.

13. Write an expression that evaluates to (not prints, not returns) the number of elements in an `ArrayList` named `list`.

14. Which of the following are benefits of object-oriented programming? Check zero or more.

    (a) Objects help organize our code, bundling code together with the data it acts upon.

    (b) Objects allow us to model the problem we are trying to solve at a high level.

    (c) Objects hide complexity.

    (d) Objects are a means of adding new, custom types to our programs.

15. What are the values of array `r` after the following code is executed?

```
String[] r = {"o", "", "", ""};
for (int i = 1; i < r.length; ++i) {
  r[i] = r[i - 1] + r[i - 1];
}
```

16. What are the values of array `r` after the following code is executed?

```
int[] r = {1, 5, 3, 6, 10, 2};
for (int i = 0; i < r.length; ++i) {
  r[i] += i % 3;
}
```

17. What are the values of array `dst` after the following code is executed?

```
boolean[] src = {true, false, true, true, false, false, false, true};
boolean[] dst = new boolean[src.length];
for (int i = 1; i < dst.length - 1; ++i) {
  dst[i] = src[i - 1] || src[i] || src[i + 1];
}
```

18. Write method `add` that accepts two arrays of `int`s as parameters. Assume the arrays have the same number of elements. It returns back a brand new `int` array. Each element in the returned array is the sum of the corresponding elements of the parameter arrays. For example, `add(new int[] {1, 3}, new int[] {10, 20})` → `{11, 23}`.

19. Write `static` method `hasTwice` that accepts two parameters: an `ArrayList` of `String`s and a `String` to search for. It returns `true` if and only if the parameter `String` appears twice in the list.

20. Write a class `Banner`, which outlaws certain words, with the following:

    - A constructor that sets up the `Banner` to report all `String`s as okay—not banned.
    - Method `ban` that accepts a `String` parameter. It bans the `String` such that calling `isOkay` on the parameter `String` will report `false`.
    - Method `isOkay` that accepts a `String` parameter and returns `true` if the parameter has not been banned and `false` otherwise.

21. Write class `Rover`, which roves across a 2D Cartesian grid, with the following:

    - A constructor that accepts the `Rover`s initial x- and y-coordinates (as `int`s).
    - Methods `n`, `e`, `s`, and `w`, which move the `Rover` one unit north, east, south, or west. For example, `n` moves the `Rover` one unit along the positive y-axis, while `e` moves the `Rover` one unit along the positive x-axis.
    - Methods `getX` and `getY` that return the `Rover`'s current coordinates.
    - Method `reset` which returns the `Rover` to its starting location.