HOMEWORK / STUDY GUIDE 2

PROFESSOR GODFREY C. MUGANDA

1. What is a data type?

2. Give two examples of Java primitive types. For each example, give some example values and some example operations of the data type.

3. How is a Java reference type different from a Java primitive type? Give a couple of examples of Java reference types, and for each example, explain what the values look like (or give an example) and also give some examples of some of the operations of the reference type.

4. What is the purpose of the `toString()` method of an object?

5. What type of value is the *hash code* of an object? Why might Java objects need to have hash codes?

6. What is a hash code *collision*, and how do such collisions occur?

7. If two Java objects are equal according to the `equals()` method, what can you say about their hash codes?

8. If two Java objects are not equal according to the `equals()` method, what can you say about their hash codes?

9. What is an *abstract* method?

10. What is an interface?

11. What is the difference between a *list* collection and a *set* collection?

12. Assume that we have a class

```
class Pet
{
  String petOwner,
  String petName,
  int weight;
}
```

(a) Override the `toString()` method for the `Pet` class so that returns a string that might look like this

```
Pet Fido owned by Sandra
```

in the case that the field `petOwner` has value "`Sandra`", `petName` has value "`Fido`", and `weight` has value 20.

(b) Suppose that two pets are considered equal if and only if they have the same owner, and they have the same name. Override the `equals(Object ob)` method of the `Pet` class so that it works accordingly.

13. How do you specify a computational problem when an algorithm is needed to solve the problem?

14. How is the size of computational problem determined?

15. How do you recognize that a step used by an algorithm to solve a problem is *basic*?

16. Give an example of a simple computational problem together with an algorithm for solving the problem. Give an example of a basic step in the algorithm, and give an example of a step in the algorithm that is not basic.

17. Define the concept of a *worst case complexity function* for an algorithm, and also an *average case complexity function.*

18. Define what it means to say that a function $f(n)$ is $O(g(n))$ where $g(n)$ is another function.

19. Prove that if $S(n) = 1 + 2 + \cdots + n$ then $S(n) = \frac{n(n+1)}{2}$.

20. Suppose that an algorithm performs n iterations. During the first iteration it performs 1 basic step. During the second iteration, it performs 2 basic steps, and so on, so that during the nth iteration it performs n basic steps. Determine the worst case complexity function of the algorithm.

21. Suppose that an algorithm performs n iterations. During the first iteration it performs 1 basic step. During the second iteration, it performs 3 basic steps, during the third operation, it performs 5 basic steps, and so on, so that during the nth iteration it performs $2n - 1$ basic steps. Determine the worst case complexity function of the algorithm.

22. Make up a class with at least three fields, and implement both the `equals()` method and the `Comparable` interface for it.

23. The `String` class implements the `Comparable<String>` interface. What can you say about the value of the integer returned by the call

(1) `"hello".compareTo("hello")` ?
(2) `"abc".compareTo("xyz")` ?
(3) `"xyz".compareTo("abc")` ?

24. Consider the class `Cord`.

```
class Coord
{
   int x;
   int y;
}
```

Two objects of this class are equal if and only if their x fields are equal and their y fields are also equal.

Define a natural order for `Coord` objects as follows. U comes before V if `U.x` is less than `V.x`, or, if `U.x` is equal to `V.x`, then U comes before V if `U.y` is less than `V.y`.

Write a method that overrides the `equals()` method for `Coord`, and modify `Coord` to implement the `Comparable` interface.

25. Refer to Exercises 1 on recursion as you study for Quiz 2! There may be questions on recursion on the quiz.