Name: Andrew Januszko

CSC110 Exam 2

Vocabulary - 1 pt each

Use the following words to fill in the blanks in this section. Note: you can use words more than once, you may need to pluralize the words, and you will not use all of the words

√ Annotation	√ Benchmarking	Best case analysis	√ Big-Oh
√ Border Case	√ Count-Controlled Loop	DeMorgan's Law	√ Early Exit Condition
√ Lifetime	Method Variable	Nested Conditional	Response Time
Scope	Selection Sort	Sentinel-Controlled Loop	√ Short-cutting evaluation
Sorting Problem	TDD	this	Worst Case Analysis

1. We use a/n Annotation to mark a method in our test class so that it will run before each of the tests in that class.
2. We focus our tests on Border Cases because that is where bugs are most likely to occur
3. The time during which a variable exists in memory is called Lifetime
4. We could measure Response Time as part of benchmarking a system.
5. We generally use a for loop to implement a Count-Controlled loop
6. Selection Sort is one algorithm that solves the Sorting Problem
7. When we need to distinguish an instance variable from a local variable with the same name, we use the keyword this.
8. If there is a return in the middle of a loop, we call that a/n Early Exit Condition
9. Sometimes, the compiler will skip the computation of part of a conditional because it knows that the value of that conditional won't change. We call this Shortcutting evaluation
10. Theoretical run-time is independent of the machine it runs on and is denoted with Big-Oh notation.

Short Answer

11. (4 points) Give two reasons why we look for Red before we start building our solution.

Helps to catch mistakes made when writing a test
To test functionality

12. (4 points) Suppose you are building a system that holds five values in an array. What are your border cases?

The beginning of the array and the end
to make sure we know where to start and that we do not walk off the end

13. (4 points) Give two examples of things we clean up in refactoring.

Clearing up unused variables
fixing java comments

14. (1 points) True or False: If you fully parenthesize your code, you don't have to worry about the dangling else problem

True

15. (1 points each) What type of statement do you use for each of these situations:

(a) choosing between two possible options boolean
(b) count-controlled loops for loop
(c) sentinel-controlled loops while loop
(d) choosing not to some one possible option if (!condition){}
(e) marking a method as a test @ test
(f) marking a method so it is run before each test in a test class @ Before

16. (4 points) Explain why scope is a compile time issue while lifetime is a runtime issue.

Variable scope is decided at compile time.
Variable lifetime when it is created and destroyed.

17. (4 points) We said that the lifetime of an instance variable is the same as the lifetime of the object that contains it. What makes that object's lifetime end?

When the class that uses the variable ends

18. (4 points) Explain why we can throw away the constants and lower order terms when we use Big-Oh notation.

When a variable gets larger we do not need the constants because of how little they change the runtime

19. (4 points) Explain why we want our run-time analysis to be machine independent.

We want our program to have the same results on every system, so we want our runtime to be independent of the machine type

Code Constructs

20. (2 points) What is the difference in how you declare an instance variable and a local variable? Local variables are declared in a method, while instance variables are delcard at the start of a class

21. (2 points each) In regards to assertEquals,
    (a) What are its parameters? (expected value, actual value, margine of error)
    (b) How does it affect the test when they are equal? The test passes
    (c) How does it affect the test when they are not equal? The test fails
    (d) Can you have more than one assertEquals() in a test? yes

Be The Machine

22. (2 points each) Mark or define the scope of every variable in this code

```
public class SillyClass
{
        private int var1;
        public int var2;

        public void myMethod(int var3)
        {
                int var4;
                System.out.println("here");
                var4 = 16;
                int var5 = 42;
                for (int var6 = 0; var6 < var5; var6++)
                {
                        var5 = var5 - 5;
                        int var6 = 42;
                        doTheThing(var5);
                }
                System.out.println("also here");
        }

        public void doTheThing(int v)
        {
                while (v > 6)
                {
                        System.out.println(v);
                        v--;
                }
        }
}
```

Sorting

23. (4 points) Which sorting algorithm would use the most swaps in the average case?

Bubble Sort

24. (4 points) Show the swaps made by Insertion Sort on this data: 4, 8, 10, 3, 6, 12

25. (4 points) Explain why Insersion Sort's run-time depends on the ordering of the data while that is not true for the other two algorithms we studied?

Because it passes to make the values to the correct place, also if it is in the right place it will skip it.

Rogue

Consider this method from our Board class :

```
/**
 * Will randomly place items in the walkable portion of our map
 *
 * @param howMany the number of items we should place
 */
void addItems(int howMany)
{
  int numberPlaced = 0;
  while (numberPlaced < howMany)
  {
    int x = (int)(Math.random()*boardWidth);
    int y = (int)(Math.random()*boardHeight);
    if (boardState.charAt(calculateBoardStringPosition(x , y)) == '.')
    {
      int type = (int)(Math.random()*ItemType.values().length);
      Item thing = new Item(ItemType.values()[type],
          "Henry",
          (int)(Math.random()*
              (ItemType.MAX_STRENGTH - ItemType.MIN_STRENGTH +1)
                +ItemType.MIN_STRENGTH),
          true);
      thing.setX(x);
      thing.setY(y);
      items.add(thing);
      boardState.setCharAt(calculateBoardStringPosition(x, y),
      thing.getType().getPrintChar());
      numberPlaced = numberPlaced + 1;
    }
  }
}
```

26. (2 points) Is the loop sentinel-controlled or count-controlled. Explain.

Sentinel, because howMany can be a random value

27. (2 points) How do we know that a position on the board is "walkable"?

We know if a position is walkable because it has the same char as '.'

28. (6 BONUS points!) Explain each of the values we passed into parameters of the constructor of Item.