COMP 1451 assumes you have passed COMP 1409 or have equivalent experience. Please answer the following questions as best you can to help me get a sense of where you're starting from. Don't worry, this is not a graded quiz. We will discuss the answers later.

OOP concepts

Use this Java statement to answer the questions that follow it.

```
Cat cat = new Cat("Garfield");
```

This statement first declares a reference variable of type Cat. The reference variable's name is cat. The statement also invokes the constructor of the Cat class, creating a new Cat object. The location of the new Cat object is assigned to the reference variable cat. It is important that you understand the difference between the reference and the object it refers (points) to. The reference is used to "get at" the object, but is not the object itself.

1. What is `cat`?
   a) A class.
   b) A reference.
   c) An object.
   The correct answer is b.

2. What is `Cat`?
   a) A class.
   b) A reference.
   c) An object.
   The correct answer is a. The class Cat specifies the datatype of the reference variable cat.

3. What does `new Cat("Garfield")` create?
   a) A class.
   b) A reference.
   c) An object.
   The correct answer is c. The Java keyword **new** is used in conjunction with a call to the constructor of the Cat class, creating a new Cat object. The string "Garfield" is the parameter to the Cat constructor.

4. In the example above, what is the difference between `cat` and `Cat`?
   As explained above, cat is a reference variable that will hold a reference to an object of type Cat. The class Cat is also the datatype of the variable, since class specifies datatype.

5. Encapsulation is one of the major concepts of OOP. What does this mean?
   Encapsulation is a technique used by OO programmers to ensure the

implementation details of an object are “hidden” inside the object. Only information about **what** a class can do should be visible to the outside, not **how** it does it. We enforce the separation between “what” and “how” by making the fields (instance variables) private and using accessor and mutator methods.

6. Abstraction is one of the major concepts of OOP. What does this mean?
   Abstraction is the ability to ignore details and focus on a higher level of a problem. Abstraction is used not just in OOP, but also in structured programming when we “abstract out” a series of instructions to create named procedures and functions which we then call as needed.

7. Trace through the Java code below and indicate what will be displayed on the screen. Start tracing from the static `mainMethod()` at the bottom.

```
public static void mainMethod() // START TRACING HERE
{
     Trace t1 = new Trace(25);
     Trace t2 = new Trace(30);
     System.out.println("t1.getX()+" "+t2.getX());
     t2 = t1;
     t2.setX(10);
     t1.setX(90);
     System.out.println(t1.getX()+" "+t2.getX());
     if (t1.getX() == t2.getX()) {
       System.out.println(“Same values!);
     }
     if (t1 == t2){
       System.out.println("Same object!");
     }
     else {
       System.out.println("Different objects!");
     }
}
```

Output

```
25 30
```

```
90 90
Same values!
Same object!
```

8. Trace through the following method and show in the space below what will be on the screen.

```
public static void traceQ()
{
    int number = 5;
    int value = number * 2;
    boolean check = false;

    while(number > 0 && !check) {
      if (value >= 20) {
          check = true;
      }
      if (value >= 10) {
          value += number;
      }
      System.out.println("value: "+value+ " number: "+number);
      number--;
    }
}
```

Output

```
value: 15 number: 5
value: 19 number: 4
value: 22 number: 3
value: 24 number: 2
```

9. Trace through the following method and show in the space below what will be on the screen.

```
public static void draw()
{  // first fill the grid
   char[][] grid = new char[4][3];
   for(int i = 0; i < grid.length; i++) {
     for (int j = 0; j < grid[i].length; j++) {
        if ((i % 2 == 0) && (j % 2 == 0)) { // even numbers
           grid[i][j] = '*';
        }
```

```
            else {
               grid[i][j] = '$';
            }
          }
        }
      // display the grid contents
      for(int i = 0; i < grid.length; i++) {
        for (int j = 0; j < grid[i].length; j++) {
           System.out.print(grid[i][j] + " ");
        }
        System.out.println();
      }
   }
```

Output

```
* $ *
$ $ $
* $ *
$ $ $
```

10. What value is printed to the screen after each of the following segments of code is executed?

```
    int x = 4;
    boolean y = ((x <= x) || (x == 5));
    System.out.println(y);
```

Answer: _**true**___________________________

```
    String a = new String("Joe");
    String b = new String("Joe");
    if(a == b) {
        System.out.println("same");
     }else{
        System.out.println("different");
     }
```

Answer: __**different**___________________________

```
System.out.println(5 + 5 + "5" + 5 + '5' + 5);
```

Answer: __**105555**_____________________________

```
System.out.println((25 / 2) > (25 % 4));
```

Answer: ___**true**_____________________________

```
System.out.println(!(5 > 4) && ((5 < 4) || (5 == 4)));
```

Answer: __**false**_____________________________