1. In class, we learned about two different types of exceptions. What are they called and how do they differ? Give an example of one situation where each exception might be thrown.

Solution: The two types of exceptions are checked and unchecked exceptions.

Unchecked exceptions are problems with the code itself that Java found while compiling, but the programmer might have missed. Examples of unchecked exceptions include NullPointerException, Array/StringIndexOutOfBoundsException.

Checked exceptions are problems that occurred outside of the code that the programmer might not have been able to foresee, since they were caused by the program user instead of the programmer. Examples include FileNotFoundException (user supplied nonexistent file), IllegalArgumentException (user supplied an argument to a class that isn’t allowed, such as giving Integer.parseInt() a string of letters).

2. Write a short java class that contains, at minimum, two instance variables of different types, one constructor, one getter, one setter, and a toString method. Make sure you use proper naming conventions and indent appropriately.

Answers may vary a lot - but here’s one correct example! To get full points on this in a test, you would need to have correct syntax, parameters, return types, and naming conventions but variable names and the toString method could vary greatly.

```java
public class SectionLeader{
    //The instance variables
    private String name;
    private int class;
    //The constructor
    public SectionLeader(String name, int class){
        this.name = name;
        this.class = class;
    }
    //The getter
    public String getName(){
        return name;
    }
    //The setter
    public void setClass(int class){
        this.class = class;
    }
    //The toString
    public String toString(){
        return "Name= " + name + ", Class= " + class;
    }
}
```
3. Select the correct answer for the question “Composition is best described as when __________?”. Then, give an explanation for what you picked:

A: An object is an extension of another object  
B: An object implements an interface  
C: An object has another object as part of its state  
D: An object inherits from another object

Answer should be C. Choices A and D generally mean inheritance, which is far more closely related to adaptation, and answer B is not the best description of composition. Answer c is the correct answer.

4. True or false: You can directly instantiate an abstract class

Answer is false. You cannot instantiate an abstract class as an abstract class must have abstract methods and those methods must be completed by the programmer before you can instantiate the abstract class.

5. Let’s say we want to inherit from two classes, the Student class and the Employee class. As you know, java does not allow multiple inheritance. Discuss some ways around not being able to inherit from multiple classes.

Answers can vary. Could include turning one (or both) of the classes to interfaces or something like chaining the inheritance hierarchy (you extend student, who extends employee).

6. Create a new PrintWriter object from a BufferedWriter object that will write to a file named “PracticeTestsAreFun.txt”

`PrintWriter writer = new PrintWriter(new BufferedWriter(new FileWriter("PracticeTestsAreFun.txt")));`
7. The code below has a problems with it. Some that might cause the program not to compile and some that are just plain bad practice. What should the programmer do to fix the code?

```java
import java.io.BufferedReader;
import java.io.FileReader;

public class TestHelper {
    public static void main(String[] args) {
        BufferedReader myReader = new BufferedReader(new FileReader("Test1_Answers.txt"));
        int input;
        while((input = myReader.read()) != -1) {
            System.out.print((char)input);
        }
    }
}
```

There are 2 things wrong. The first thing is to add a try catch around the read statement and the FileReader constructor. Making main throw IOException works as well since we are just getting it to compile and run. Lastly, myReader is not closed.

8. We know about the method substring in the Java String class. We can call this method in two ways with either 1 int argument or 2 int arguments. This is an example of what?

A. Overriding  
B. **Overloading**  
C. Adapting  
D. Composition

9. Below you are given addition and subtraction in decimal number. Write the answer in binary and in decimal. Please use an 8-bit value number for your answer

a. 127 - 80  
b. 127 - (-127)  
c. -80 - 5  
d. -31 - 100  
e. 100 + 32  
f. 54 + 12
<table>
<thead>
<tr>
<th></th>
<th>Binary</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>00101111</td>
<td>40</td>
</tr>
<tr>
<td>b.</td>
<td>11111111</td>
<td>-1</td>
</tr>
<tr>
<td>c.</td>
<td>10101011</td>
<td>-85</td>
</tr>
<tr>
<td>d.</td>
<td>01111101</td>
<td>125</td>
</tr>
<tr>
<td>e.</td>
<td>10000100</td>
<td>-124</td>
</tr>
<tr>
<td>f.</td>
<td>01000010</td>
<td>66</td>
</tr>
</tbody>
</table>

10. What is the difference between an ArrayList and an array?

Answers may vary, but here are a few of the correct answers: You cannot change the size of an array once it is made. However, an ArrayList's size can be changed indefinitely. One is static (array), while the other is dynamic (arraylist). ArrayLists are objects while arrays are not.

11. Is ArrayList a...?

a. **Generic Class**
   
b. Abstract Class
   
c. Interface
   
d. Abstract Data Type

12. Define ADT (Abstract Data Types) and list one example of an ADT.

An ADT consists of data and operations on that data, specified independently of a programming language. An example of an ADT that we went over in class is a list.

13. Give the definition of the following:

   a. **Class Composition**: An object whose state includes other objects exhibits composition
b. **Class Adaptation:** A class that adapts an existing class uses composition to provide a new set of methods to manipulate objects of the existing class.

c. **Class Inheritance:** Inheritance permits a new class to be created with the variables and methods of an existing class as the foundation on which additional variables and methods may be added.

d. **An Interface:** Has no code. No methods are implemented. Completely separated from the implementation.

14. Create an example interface that could be used for some methods in Super Mario World! (or any another game you know) Your interface should have at least three methods: One that returns a int, one that returns a boolean, and one that returns an array or list.

*Many, many different answers are possible. Here’s our answer, but any answer that includes three methods with complete method signatures, the correct return types, and no code inside of them would be correct.*

```java
public Interface SuperMario{
    public boolean isDead();
    public int lives();
    public String[] getLevelName();
}
```
15. What is stored in result?
Stack stck = new Stack()
stck.push('a');
stck.push('b');
stck.push('c');
stck.pop();
Char result = stck.pop()

Answer: 'b'

Note: stacks may or may not be on the test, depending on how much time each class has to talk about stacks before the exam, but that doesn’t mean you can’t study up on them anyway!

16. How are the end of lines noted in UNIX text files? DOS text files?
UNIX \n
DOS \r\n
17. Why would you want to encapsulate a class?
To separate how a class is implemented (code and data) from the functionality of the class.
Or to hide the code and data of a class from a programmer who is using the class.

18. Given the following code and text file “example.txt”, write the correct output:
example.txt:
Hello
World!
Sup
BufferedReader br = null;
try
{
    br = new BufferedReader(new FileReader("example.txt"));
}
catch (FileNotFoundException e)
{
    System.out.println("ERROR: File example.txt not found.");
}
System.out.println(br.readLine());
System.out.println((char)br.read());
System.out.println((char)br.read());
Output:
Hello
W
0

19. Initialize the following:
   A string called “testString” that contains the word “hello” with a new line afterwards

   ```java
   String testString = "hello\n";
   ```

   An integer called “rand” that stores a random int from 0, inclusive to 10, exclusive

   ```java
   int rand = (int) (Math.random() * 10);
   OR
   Random rng = new Random();
   int rand = rng.nextInt(10);
   ```

   A two dimensional array called “coords” that stores integers with the size of the first dimension being 5 and the size of the second dimension being 10.

   ```java
   int[][] coords = new int[5][10];
   ```

   Now, write a snippet of java code that will fill the “coords” 2D array with random integers between 1 and 10.

   ```java
   for(int x = 0; x < 5; x++) {
       for(int y = 0; y < 10; y++) {
           coords[x][y] = (int) (Math.random() * 10);
           OR
           Random rng = new Random();
           coords[x][y] = rng.nextInt(10);
       }
   }
   ```
20. Name 2 number types in Java and their equivalent wrapper classes

Two possible answers: int and Integer, double and Double

21. What is the difference between a static variable and a non-static variable?

Answer: A non-static variable exists only in relation to an instance of the class but static variables exist throughout the lifetime of the class (put simply, you don’t have to call new before you can access a static variable)

22. If x has already been declared as an integer, what would the output be when this statement is executed when x = 0 and when x = 1?

assert x == 1: “Oh nooooooooo!!!!!!”

Answer: If x is 1, nothing gets outputted. If x is 0, java throws the AssertionError and “Oh nooooooo!!!!!” is printed along with the stacktrace.

23. Complete the following java class:

public class ArrayListsRock {
    private ArrayList<String> myList;

    public ArrayListsRock(){
        //Initialize the ArrayList but do nothing else!

        myList = new ArrayList<String>();
    }
}
public void makeFirst(String first) {
    // Set the first element in the arraylist to first
    // shifting the other elements down
    myList.add(0, first);
}

public void makeLast(String last) {
    // Set the last element in the arraylist to last
    // without changing any of the previous elements
    myList.add(last);
}

public void changeAtIndex(int index, String change) {
    // Replace the element at index with change
    myList.remove(index);
    myList.add(index, change);
}

public String toString()
{
    // Print out the contents of the ArrayList, with one string on each line

    String str = "";
    for (int i = 0; i < myList.size(); i++) {

        str = str + myList.get(i) + "\n";
    }

    return str;
}