Short answer

Answer the following questions with a short and concise answer. You should not need more than a sentence or two for each answer.

1. (4 points) There are literally hundreds if not thousands of classes defined in the java class hierarchy; They all have one superclass in common.
   (a) What is the name of this class?
   (b) Name one method defined in this super-ancestor. (What is a method you could call on any object instance?)

2. (4 points) If you have not written a default (parameter-less) constructor for your class, but you have written a constructor that takes at least one parameter, can you still call the default constructor on that class? Why?

3. (4 points) Object oriented programming languages such as Java has many important features, however one of the more important ones is the distinction between a class and an object. Please explain the difference briefly.
4. (4 points) What are two differences between an abstract class and an interface?

5. (4 points (bonus)) When would it make sense to have a private constructor in a class?

Access Modifiers

6. For each part of this question, write all choices that apply on the answer line.
   A. the class in which it is declared.
   B. classes that extend the class in which it is declared.
   C. parent classes of the class in which it is declared.
   D. classes in the same package as the class in which it is declared.
   E. classes nested inside the class in which it is declared.
   F. all other classes.
   (a) (2 points) A member with declared with a public access modifier is visible to:
      
   (b) (2 points) A member with declared with a private access modifier is visible to:
      
   (c) (2 points) A member with declared with a protected access modifier is visible to:
      
   (d) (2 points) A member with declared without an access modifier is visible to:
Inheritance

7. Consider the following code.

```java
public class Parent {
    public int x = 0;

    public Parent(int x) {
        this.x = x;
    }

    public class InnerChild extends Parent {
        public int x = 1;

        public InnerChild(int x) {
            super(2*x);
            this.x = x;
        }

        public void innerMethod(int x) {
            outerMethod(x);
            System.out.println(x);
            System.out.println(this.x);
            System.out.println(super.x);
            System.out.println(Parent.this.x);
        }
    }

    public void outerMethod(int x) {
        System.out.println(x);
        System.out.println(this.x);
    }

    public static void main(String[] args) {
        int x = 37;
        Parent p = new Parent(x);
        x = 6;
        Parent c1 = p.new InnerChild(x + 1);
        InnerChild c2 = p.new InnerChild(x - 1);

        p.outerMethod(2);
        c1.outerMethod(3);
        c2.innerMethod(4);
    }
}
```
This page intentionally left (mostly) blank so the code and questions about it will be on separate sheets of paper.
(a) (10 points) What is the output of this code?

(b) (6 points) Which of the following lines of code would cause an error when placed in the main method of Parent after the code that is currently there? Circle all that apply.

A. p.innerMethod(5);
B. ((InnerChild)p).innerMethod(5);
C. c1.innerMethod(5);
D. ((InnerChild)c1).innerMethod(5);
E. Parent p2 = new Parent(10);
F. Parent p2 = new Parent(10).new InnerChild(20);
G. Parent p2 = p;
H. Parent p2 = c2;
I. InnerChild c3 = new Parent(10);
J. InnerChild c3 = new Parent(10).new InnerChild(20);
K. InnerChild c3 = p;
L. InnerChild c3 = c2;
M. InnerChild c3 = c2.new InnerChild(20);
8. Consider the following inheritance structure.

\[
\begin{array}{c}
\text{ComicCharacter} \\
\text{\hspace{1cm} speak(): void} \\
\end{array}
\]

\[
\begin{array}{c}
\text{Hero} \\
\text{\hspace{1cm} saveTheDay(): void} \\
\end{array}
\]

\[
\begin{array}{c}
\text{Killable} \\
\text{\hspace{1cm} isDead(): boolean} \\
\end{array}
\]

\[
\begin{array}{c}
\text{Villain} \\
\text{\hspace{1cm} plotEvilScheme(): void} \\
\end{array}
\]

\[
\begin{array}{c}
\text{SupaWoMan} \\
\end{array}
\]

\[
\begin{array}{c}
\text{LexxiLoother} \\
\end{array}
\]

(a) (5 points) We are allowed to create the method `speak()` in the abstract class `Hero` even though it’s already been defined in the `ComicCharacter` class.

i. Is this practice called overloading or overriding?

ii. Explain what the other concept (the one that you didn’t choose) refers to.

(b) (5 points) Please write the code necessary for two new classes - one in the Hero category, and one in the Villain category. You may assume that the `saveTheDay` and `plotEvilScheme` methods have already been implemented in the abstract classes. Also, add them to the class diagram above.
(c) (5 points) Most super heroes come bundled with a side kick, though all of them are usually a bit more vulnerable than the hero, so they can all definitely be killed. Please add the `SideKick` abstract class to the class diagram above with the appropriate inheritance arrows, and explain why you drew what you did.

(d) (5 points) In addition to having side kicks, most heroes have an arch enemy that usually try to destroy the world as we know it. So...it would make sense to declare a `protected Villain archEnemy` instance variable in the `Hero` class. However, someone might argue that each `Villain` instance should also have a `protected Hero archEnemy` instance variable declared since the association is always both ways. Then however, wouldn’t it be good to put `protected ComicCharacter archEnemy` as an instance variable in the abstract class `ComicCharacter` instead? Please give at least one pro for each approach.

(e) (2 points (bonus)) Suggest one more method for each of the two classes `Hero` and `Villain`. Name the methods and suggest return types and arguments.
Exceptions

9. (5 points) You can have multiple catch clauses for each try clause. Why is the order of the listed catch clauses important?

10. (5 points) Here is the inheritance structure for a few classes (the $\rightarrow$ means “is extended by”):

```
java.lang.Object
    $\rightarrow$ java.lang.Throwable
    $\rightarrow$ java.lang.Exception
        $\rightarrow$ java.lang.RuntimeException
        $\rightarrow$ java.lang.IllegalStateException
        $\rightarrow$ java.lang.NullPointerException
    $\rightarrow$ java.io.IOException
        $\rightarrow$ java.io.FileNotFoundException
        $\rightarrow$ java.io.IOException
```

Which of the above seven (7) exception classes are checked exceptions, and which ones are unchecked exceptions? What’s the difference between the two types of exceptions? Give one more example of an unchecked exception that we’ve talked about in class.
11. This question refers to the classes from question 8.

   (a) (5 points) For future use it might be good to have a SuperHeroException class prepared. Please write the code for this custom made exception class.

   (b) (5 points) Suppose the SupaWoMan class declares a method

       public void chaseVillain ( Villain v, int time )

   where v is the villain that we’re chasing, and time is the time in minutes we’ll let our hero chase the villain of choice. Show what changes need to be made to the chaseVillain method header in order to allow this method to cause a SuperHeroException. Also, show the code that would generate this exception if the chase time was reached without catching the villain.
(c) (5 points) Assume that in addition to the ComicCharacter class hierarchy, we also define exceptions in the same structure:

```
ComicCharacterException
  +-> SuperHeroException
  +-> VillainException
  +-> SideKickException
```

Then, if I wrote a test driver like so:

```java
ComicCharacter villain = new LexxiLoother();
ComicCharacter hero = new SupaWoMan();
try {
    villain.methodThatMayCauseException();
    hero.methodThatMayCauseException();
} catch (ComicCharacterException cce) {
    // This is the place I'm interested in
}
```

Now, if either of the two hero/villain methods cause an exception, they will both be caught by the only catch clause I have. The question is, how do I figure out if the exception was caused by my hero or by my villain? (In other words, how do I know if the ComicCharacterException that I caught is a SuperHeroException or if it is a VillainException?)
(d) (5 points (bonus)) Assume the constructor for the LexxiLoother villain contains code for loading an initial evil plot from a text file - it might look something like this:

```java
String initialPlot = "";
File inputFile = new File("/home/lexxiloother/initialplot.txt");
Scanner sc = new Scanner(inputFile);
while (sc.hasNextLine()) {
    initialPlot += sc.nextLine();
}
```

Unfortunately this is risky business and a lot of things could go wrong.

- The `File` constructor can cause a `NullPointerException` if passed null.
- The `Scanner` constructor can cause a `FileNotFoundException` if the file the `File` object refers to doesn’t exist.
- The `hasNextLine()` method can cause an `IllegalStateException` if the `Scanner` has been closed prematurely.
- The `nextLine()` method can cause either an `IllegalStateException` if the `Scanner` was closed, or a `NoSuchElementException` if there isn’t another line to be found in the scanner.

So... Please write rewrite the above code with the appropriate try/catch blocks inserted in the right places so that I can compile my program. (Hint: you may want to take another look at the class hierarchy in question 10 before you start working on this question.)
GUIs

12. (5 points) Java uses layout managers to keep track of where components added to a GUI are being displayed. (You used some for Part 2 of the Tetris project.) Please pick a layout manager class and describe how components are laid out within it.

13. (5 points) Often times the default layout managers are not sufficiently adaptive to make the GUI that you desire. To fix it we can use “composite layouts”. Please explain how to make a composite layout.

14. (5 points) There are three ways that we can create an ActionListener and add it to a button. Please pick one way and explain how to create the listener that way. Also specify one benefit, and one drawback of creating the listener for the approach you choose.