Questions About Classes
Here is a rudimentary class:

```java
public class Person {
    private string name;
    private int age;
    ....
}
```

In some other class I construct a person with
```
Person x = new Person( "bob" )
```

Could I then say `x.age=69`?

A. Yes
B. No, but I could if I removed "private" from the declaration of age.
C. No, but I could if I changed the declaration of age to "public int age".
D. No, people aren't allowed to be that old.
Answer C: No, but I could if I changed the declaration of age to "public int age". But this is still a bad thing to do. We will often access private variables of a class through "getters and setters". In other words, the class will contain public methods

```java
public int getAge() {
    return age;
}
```

```java
public void setAge(int a) {
    age = a;
}
```

To set the age of x we'll say `x.setAge(69);`
Note that answer (B) in the first question (just removing the word “private”, giving us default access) would allow other classes in the same package to access variable age, but not classes in other packages. This still isn’t a good solution; use getters and setters.
I have a class Critter that has constructor
   public Critter(String species);
and a method
   public int numLegs();
What code can I put in public static void main(String [] args) to test this?

A. System.out.println( numLegs );

B. System.out.println( numLegs() );

C. System.out.println( new Critter("cat").numLegs() );

D. Critter x = new Critter("cat");
   System.out.println(x.numLegs());
Answer: C and D both work, though C is a bit odd -- it constructs a new cat, prints its number of legs, and then throws out the cat it constructed.
What is wrong with this program? The line with the red arrow has an error message that says “Unhandled exception type: FileNotFoundException”.

package examples;
import java.io.File;
import java.util.Scanner;

public class Tester {
    public static void main(String [] args) {
        String fileName = args[0];
        Scanner scan = new Scanner(new File(fileName));
        while (scan.hasNextLine()) {
            System.out.println(scan.nextLine());
        }
    }
}

A. The line with the red arrow should say
    Scanner scan = new Scanner(new File(args[0]));
B. The file isn’t where Eclipse thought it would be.
C. The file name in the Run Configuration is misspelled.
D. We need a try-catch block around the line with the red arrow.