SYSC 2004
Winter 2018
Lab 4

Objectives

1. Introductory exercises in inheritance

Submission Requirements: (Exact names are required by the submit program)

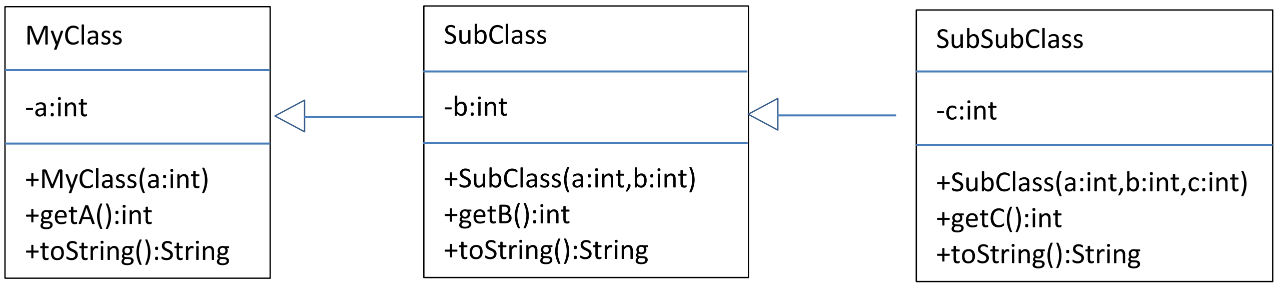
- **Without a submission you will not get a mark**
  MyClass.java, SubClass.java and SubSubClass.java, and SubSubClassTest.java

The classes that you will develop have no meaningful use or purpose. The complete focus is on understanding the visibility of members and methods along the inheritance hierarchy and the order of execution when using inherited and overridden methods.

Part 1 – Subclasses that (mostly) Inherit and Extend

Look at the three classes shown.

- Look at the instance variables:
  - `MyClass` has one instance variable
  - `SubClass` has one instance variable and one inherited (and inaccessible) variable
  - `SubSubClass` has one instance variable and two inherited (and inaccessible) variables
- Look at the constructors:
  - `MyClass` has one argument,
  - `SubClass` has two arguments and
  - `SubSubClass` has three arguments
- Look at the methods:
  - `SubClass` inherits getA(), extends getB() and overrides toString()
  - `SubSubClass` inherits getA() and getB(), extends getC() and overrides toString()



For this first part, we are going to proceed as normal: You will implement the classes according to the UML and will test your implementation against my JUnit test code.

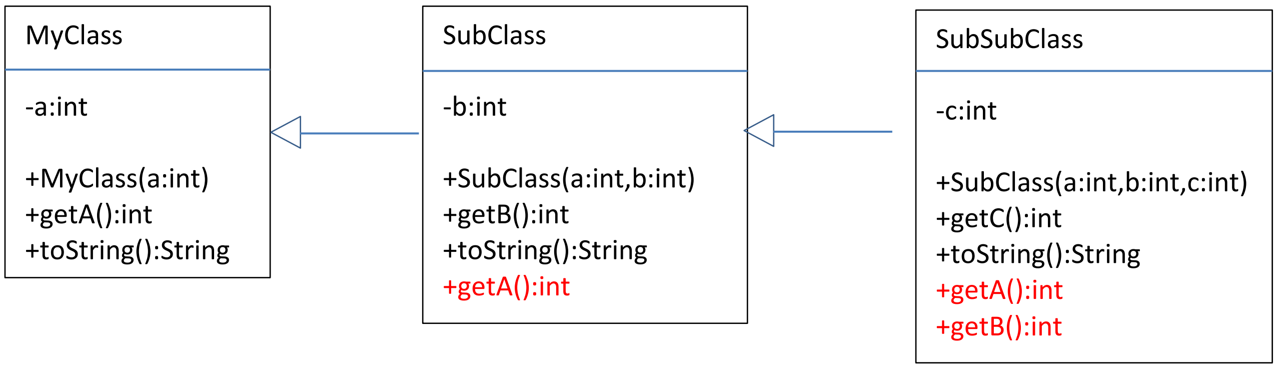
1. Create a project called **lab4Part1**.
2. Create the three classes shown in the UML above.
   - Notice how the constructors in the sub-classes chain the constructors of their parent classes, using `super`(…)
   - Notice how the `toString`() method in the sub-classes append their information onto the string information returned by their parent class's `toString`() method.
3. Test your implementation of the three classes, using the provided test code **SubSubClassTest-Part1.java** (We will test all three classes from the bottom child class).
   - Either: Right-Click on project and select New->JUnit test
   - Or (if you don't see it): Right-Click on project and select New->Other followed by "Unit Tests" for Categories and "JUnit Test" for File Types.
   - Download the test code (**SubSubClassTest-Part1.java**) from the CULearn page.
   - The test code creates three objects, called `object`, `subObject` and `subsubObject`, each one of the respective class.
4. When you have correctly passed all tests, I want you to do something you've never done before: Modify the test code for `SubSubClassTest`.java
   - Add the following tests at the bottom of the test. It won't compile. **Why?**

```
Assert.assertEquals(7, object.getB());
Assert.assertEquals(7, subObject.getC());
```

   - Comment out these wrong tests so that the compiler errors go away. Save.
5. Submit these files, including your modified **SubSubClassTest**.java

Part 2 - Subclasses that Override

You're now going to modify the two subclasses so that they have methods that override the methods in their parent class(es).



.

1. Create a separate copy of your project called **Lab4Part2**
    - **Replace the current version of SubSubClassTest.java with the new version provided with the posted files, called SubSubClassTest-Part2.java**
2. Add the methods marked in red to your previous code using the following instructions to fill in the details of their implementation.
    - `SubClass::getA`() method should return the value of `a` multiplied by 2.
    - `SubSubClass::getA`() method should return the value of `a` multiplied by the value of c.
        - This description can be interpreted in two ways. Study the new test code.
    - `SubSubClass:getB()` method should return the value of `b` multiplied by the value of `c`
3. Run these new methods against our new JUnit test class to make sure that they work.
4. I can't force you to do this, but truly, do the math and figure out where along the inheritance tree that all the values are coming from
5. Submit your files from this second project.
    - Yes, they will overwrite your previous submission.

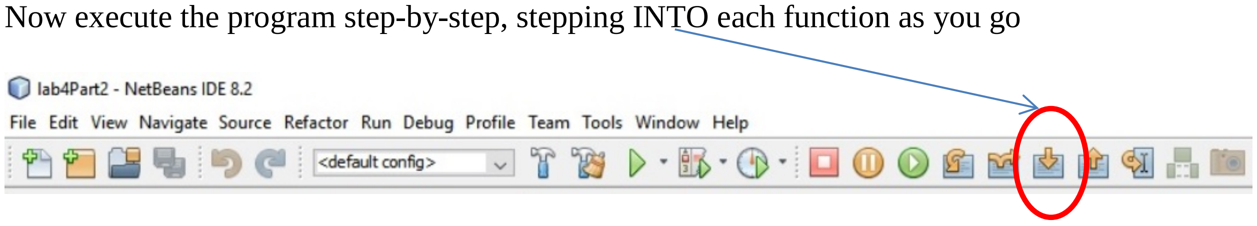
Part 3 – Debugging Inheritance

- **There is nothing to submit for this part and I can't force you to do this, but it is the best study tool for learning inheritance.**
- **Repeat the exercise for both versions that you've developed**

1. Add the provided client code called **Lab4.java** to your Source Packages. (Not your Test packages)
   a. Depending on the package name that you've used in your projects, you may have to add or remove a package statement at the top of your file to make things compile.

   Example: package lab4;

2. Look at the code. It is client code that does the exact same thing as the JUnit tests that you've run.
3. On Line 27, set a breakpoint by clicking on the line number on the right side of the code window (the first line after the declaration of value). You should see a little red square.
4. Right-click on **Lab4.java** and select -> DEBUG (NOT RUN)
5. The program will run and then stop at your breakpoint. You should see a little green error on top of your little red square.
6. Now execute the program step-by-step, stepping INTO each function as you go


lab4Part2 - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>

7. Watch your program execute as it travels up and down the inheritance hierarchy.
8. Hit the RED SQAURE in the menu bar above to terminate (and restart) your debugging session.

.