Lecture 4

JUnit
Compilation Steps

Some slides were borrowed from Josh Hug and Adam Jundt

Testing

Black-box Testing

- You don’t know (or you pretend you don’t know) how something is implemented
- You test only based on inputs and outputs

Clear-box Testing (asa “white-box testing”)

- If you can look inside the black box and see how a method is implemented, you can do more detailed testing

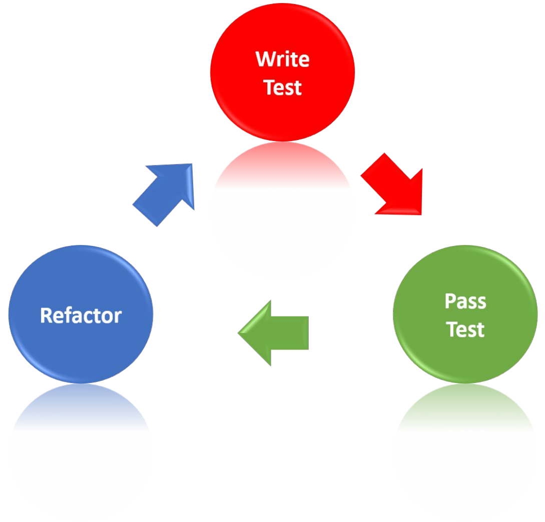
Why bother??

http://stackoverflow.com/questions/10858990/why-use-junit-for-testing

Unit (micro)-testing

- Whether you are doing black-box or clear-box testing, you should test every unit of a software system.
- What is a *unit*? In object-oriented programming, usually a software unit is taken to be a single method.
- So: we should test every method of every class in the software.
- JUnit is a widely used framework for unit testing of Java software.

Test-Driven Development (TDD)



Steps to developing according to TDD:

- Identify a new feature.
- Write a unit test for that feature.
- Run the test. It should fail. (RED)
- Write code that passes test. (GREEN)
    - Implementation is certifiably good!
- Optional: Refactor code to make it faster, cleaner, etc.

Not required but testing is always needed.

Problem to solve.

- Swap characters in the string:
    - “Marina” -> “aniraM”
    - “Class” -> “ssalC”

Idea?

Did everything make sense?

A: Yes

B: Almost, need to practice now

C: More or less, need to review (and then practice)

D: Mostly lost

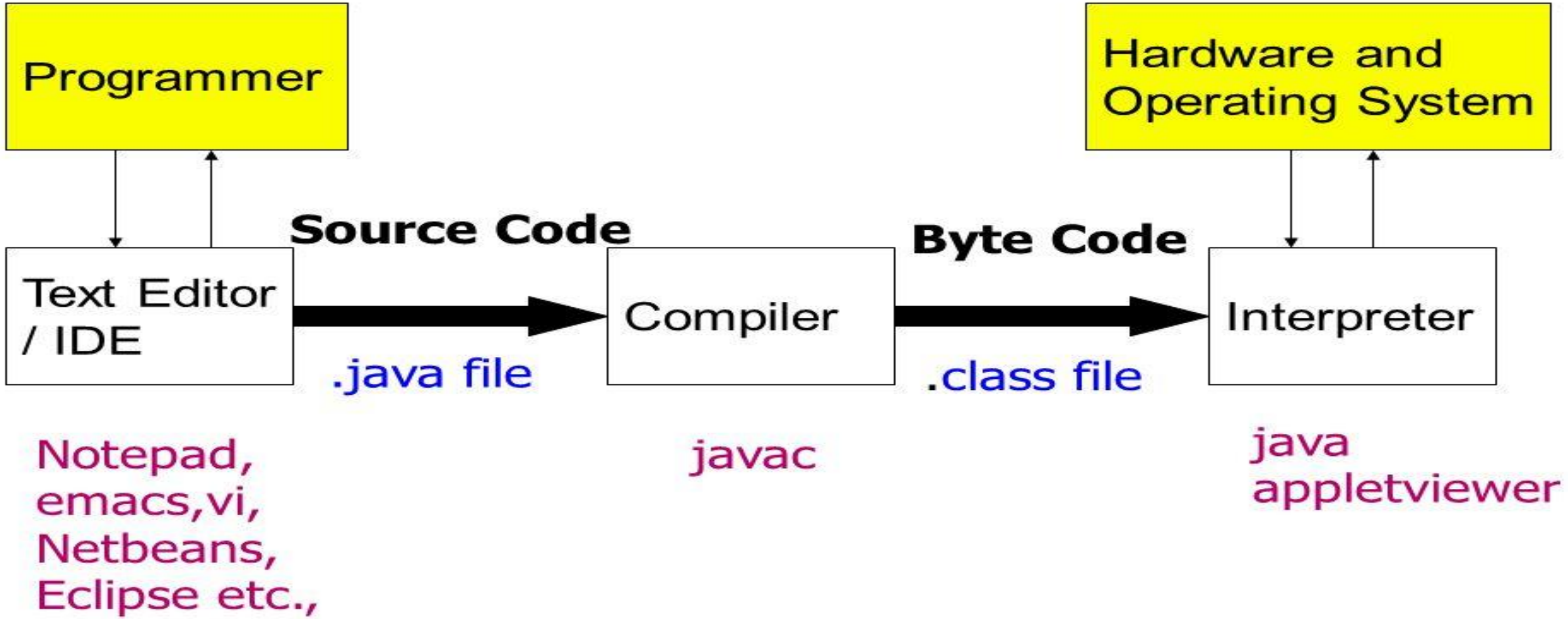
E: Completely lost

@BeforeEach, @AfterEach, @beforeAll

- @BeforeEach
  - annotation is used to signal that the annotated method should be executed before each invocation.
- @AfterEach
  - is used to signal that the annotated method should be executed after each @Test.
- @BeforeAll
  - runs once before any of the test methods in the class.

Compilation (demo)

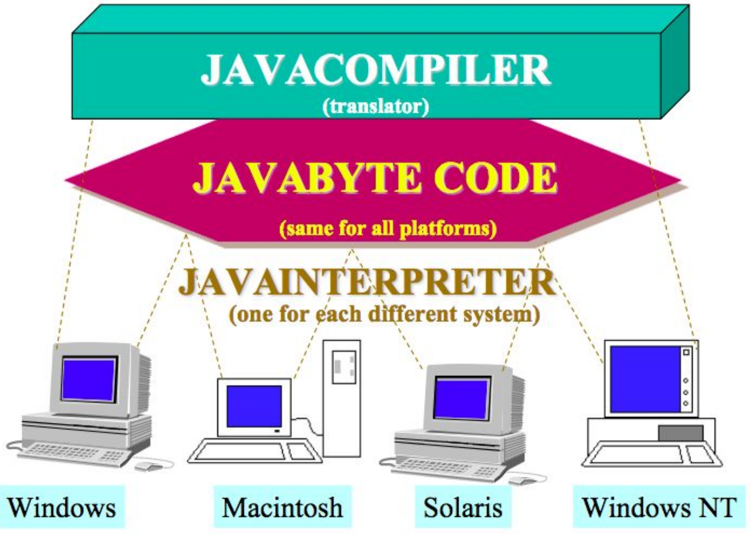
JAVA Compilation and Interpretation



Why bother with class file?

Why make a class file at all?

- `.class` file has been type checked. Distributed code is safer.
- `.class` files are 'simpler' for machine to execute. Distributed code is faster.
- Platform independent.



Instantiating the simplified `Car` class

```
public class CarLauncher {
   public static void main(String[] args){
      Car audi;
      new Car(40);
      audi = new Car(50);
     Car honda = new Car(4);

      audi.blowUp();
     honda.blowUp();
   }
}
```

```java
class Car {
     int speed;
     public Car(int sp){
       speed = sp;
     }
     public void blowUp() { System.out.println("baaaam!: " + speed);
     }
}
```

```java
public class CarLauncher {
    public static void main(String[] args) {
      Car audi;
      new Car(40);
      audi = new Car(50);
      Car honda = new Car(4);

      audi.blowUp();
      honda.blowUp();
    }
}       //Link
```