CS2110
Object Oriented Programming And Datastructures
Goals for Today

- Finish up the discussion on Types
- Introduction to Classes and Object
- Gentle Introduction To Java Testing
- Discussion:
  - Set up IntelliJ
  - Turn in the exercises on Types
  - Debug test failures in the IntelliJ IDE
  - Work Time for A1
Types Review

**Primitive Types**
- Built into Java
- Lower-case names (int)
  - Can’t be extended
  - Can’t make new ones
  - Stored efficiently
  - Matches CPU instructions
  - Building blocks of classes

**Classes**
- Names start with capital letter (String)
  - User-defined types
  - Complex operations
Types Review

- Some types can be explicitly casted to other types

```
<table>
<thead>
<tr>
<th>narrow</th>
<th>wider</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>short</td>
</tr>
<tr>
<td>int</td>
<td>long</td>
</tr>
<tr>
<td>float</td>
<td>double</td>
</tr>
<tr>
<td>char</td>
<td>int</td>
</tr>
</tbody>
</table>
```

- Functions/Operators can be overloaded to support multiple types

```plaintext
int     abs(int a) { ... }
long    abs(long a) { ... }
double  abs(double a) { ... }
float   abs(float a) { ... }
```
Introduction To Classes

- These are things that are not primitive types.
  - Some are given by Java like Class Object, Class String etc.
- But you can also define your own classes
- Every piece of java code you can think of goes inside a class ....
Example of A class

class Counter {
    int counts = 0;

    int getCount() {
        return counts;
    }
    void increment() {
        counts += 1;
    }
    void reset() {
        counts = 0;
    }
}

fields (state)

methods (behavior)

no more `static`
Classes vs Objects

- A **class** defines a *type* by specifying state and behavior
- Values of the type are called **instances** of the class
- Instances of classes are called **objects**

- Analogy: classes are **blueprints** for objects.
- An object is like a house built according to those blueprints

- Duality: classes are defined at **compile-time**, objects exist at **runtime**
Example Class: A Counter

- We define a counter here, but all you need to know is that it’s a blob of data state with some associated behaviours that go along with it.

```
<table>
<thead>
<tr>
<th>Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>• counts: int</td>
</tr>
<tr>
<td>• getCount(): int</td>
</tr>
<tr>
<td>• increment()</td>
</tr>
<tr>
<td>• reset()</td>
</tr>
</tbody>
</table>
```
More Terms: Fields and Methods

Essentially, every class has two components, its fields (aka data) and methods (procedures or behavior) that are associated with it.

- **Fields** are variables that will live inside of objects
  - Aka “attributes”, “properties”, “member variables”
  - Initialized when an object is created

- **Methods** are functions that can access an object’s fields
  - Aka “member functions”
Demo Time: Using Classes, Creating Objects

new-expression

Counter c;
c = new Counter();

Method invocation

int n = c.count();
c.increment();
c.reset();
There are things that go in classes, but are not associated directly with object specific state.

- There are identified by the keyword `static` (applies to both fields and methods)

Remember public `static` void `main(String[] args) {}` ?
Java Testing For Dummies

There are two ways to run things in Java

- Code in the main method
- Code as a test method (invoked by java’s testing framework) [Demo the discussion for this]
Discussion Time