Java

- The basic language looks much as you would expect from other languages:
  - If, else, for, while, x = x + 1; …
  - Braces { } around blocks of code
  - Functions: as in
    ```java
    int areaOfCircle(double r) {
        return Math.PI * r * r;
    }
    ```

- Variable declarations:
  - `int k;` (other primitive types: `double`, `char`, `byte`, …)
  - `Circle c;` (where Circle is a class)
  - `double[] b;` (b can contain an array of double values)

- Exceptions: “thrown” when something bad occurs (like dividing by zero or indexing off the end of an array). There is a way to “catch” such events.

A first surprise

- In some languages, allocation is “automatic”
  ```java
double[] b;
Circle c;
```
- ... not in Java. These declarations created two usable variables, but neither is initialized.

- In Java, object creation is explicit:
  ```java
  b = new double[100];
c = new Circle(2.7651);
  ```

Hello World!

```java
/** A simple program that prints Hello, world!
 * @author ken
 */
public class myClass {
  /**
   * void main(String[] args): Starting point for my program
   * @param args Command line arguments
   */
  public static void main(String[] args) {
    // At this point main is executing. We'll just print “hello”
    System.out.println("Hello, world!");
  }
}
```

An introduction to Java and objects

- Some Java features seen in our example:
  - Packages contain classes that contain code
    - Here java.lang is such a package, which is always accessible
    - As noted, normally you need to “import” each package, but not java.lang
  - Variables always have specified “types”
    - For example, parameter arg of main has type String[]
    - Use arg.Length is the length of the vector
    - In Java, you are forced to be very precise about what kinds of value each variable can hold
An introduction to Java and objects

- Some Java features already seen in our example:
  - Packages contain classes
  - Variables have specified “types”
  - A type is a set of values together with operations on them
  - Methods like main and println
    - main is static method of class myClass
    - out is a static variable of class System
    - Method out.println is a method associated with object out.
    - Eclipse knew which methods object out contains because it knows the type of out, which is PrintStream

More things to notice

- Lots of curly braces
- Indentation reflects program structure
- Eclipse is your helper in code development
  - When you start to type, it guesses what you are trying to do and offers to fill things in
  - Eclipse knows about the methods you can use in the objects you are working with
  - And it can automatically help with things like indentation, although you can override its help

Java won’t notice certain mistakes

- What does this do?

```java
n = 0;
while(n < myVec.Length && myVec[n] >= 0);
n = n+1;
```

- ... this code is an infinite loop, stuck on the “while” stmt. Java doesn’t pay attention to indentation...

What are “classes”

- In Java we distinguish between two kinds of variable
  - Variables declared with “primitive” types like int, double. Notice the lower-case type names.
  - Variables that contain a pointer to an object, or might contain null if no assignment has been done yet.
- A class defines a new type of object
  - It tells the value of the object, the operations you can do on it, how to initialize a new instance, etc.
  - We use upper-case names for class types

Auto-boxing, unboxing

- Integer is what we call a “wrapper” class:
  - It is predefined as a class containing an int value field
  - All the normal int operations work, but an instance of Integer is an object, so object operations also work.
  - In contrast, an int is a base type, not an object
- Autoboxing/unboxing
  - In any situation where you need an int, Java will allow an Integer. It automatically “auto-unboxes” the int
  - In a situation where you need an Integer and supply an int, Java will “auto-box” the int by creating an Integer object
What's in x before the new?

- The answer depends on where we declared the variable.
  - If x is a local variable, it was undefined before the first value was assigned to it. Trying to access an undefined variable is illegal; your code won't compile.
  - If x is a field in a class, its initial value is null.
    - Means that x doesn't point to an object instance
    - If you try to reference a component of x (e.g. x.value), you get an exception and the program crashes.

Reference variables “point” to objects!

- A reference variable can't point to an object until you assign an object reference to it!
- All objects are created using the new-expression.

Classes and object instances

- A class defines a type of objects.
  - Java predefines some classes, like System and Integer
- Each object has a type, which is the name of the class that was instantiated to create it
- Create an object using the new-expression, as in:
  ```java
  Integer x = new Integer(77);
  Toy t1 = new myLittlePony("Apple Bloom");
  ```

What's in an object of a class?

- An object contains fields
  - These are variables that live within each object
  - Each variable has a type, a name, and an initial value
  - We can control access
    - **private** field: can be accessed only inside the class
    - **public** field: can be accessed from outside the class.
  Software engineering principle: Make most fields **private**. Reason for the principle will become clear as the course progresses.

What's in an object of a class?

- An object also contains methods
  - Functions (return values) or procedures (do something but return nothing, indicated by void)
  - There is a way to associate them with operators
    - For example you could define “+” to call “Add”...
  - You can define the same function name multiple times with different parameter types
What's in an object of a class?

- Every class has a “constructor”
  - A method with the same name as the class
  - Its job is to initialize the class variables
  - If omitted, Java puts in this constructor:
    ```java
    public class-name() {}
    ```
  - It has no parameters. It does nothing, but very fast.
- Expression `new C(args)`
  - 1. Create object of class C somewhere in memory
  - 2. Execute constructor call C(args)
  - 3. Result: a pointer to an initialized “C” object

  Note: earlier we used the notation `C@0x17610` for such pointers

Some slightly fancy things

- We like “getter” and “setter” methods
  - `public` field: assignments to it could break the logic
  - `private` field: provide public methods to get/set the value of the field and check consistency
- Set/Get methods ensure that invariants are maintained.
  - Suppose class Circle has fields:
    ```java
    private double radius, circum;
    ```
  - Make sure they are non-negative
  - Maintain the invariant `circum = 2*pi*radius`
  - If the fields were `public`, mistakes might sneak in

Let's create a simple demo program

- Count the number of lines and characters in the file
- Fancier: could have it count the words, or make a list of words and the number of occurrences of each, or even short phrases.
- We could use this to do cutting edge research, answering questions like: When was “wrong in so many ways” first used?
- We'll use a predefined class FileStream
  - I found it using “Google” but focused on the information from Java.Oracle.com
  - Once I found it, I decided to reuse this existing class rather than try to build one of my own.
  - In CS2110, using prebuilt Java technology is encouraged but we limit ourselves to Java.Oracle.com

(Switch to Eclipse now)

Ken posted some little videos of him running Eclipse to solve a little sample problem