The Java Programming Language

- A programming language that you can use to build almost anything you can build in another language.
- A development environment: a set of tools for creating your program - such as the compiler, interpreter, documents generator, etc.
- A deployment environment: the JRE provides an easy way for deploying your program in almost any environment.
- Java's main features are:
  - The Virtual Machine (VM) – The virtual machine emulate a standard environment in a real machine. This, in turn, provides platform independence for your program.
  - Garbage collector – The garbage collection process is responsible for allocating, and de-allocating memory during the program cycle.
  - Fully Objected Oriented Language – In Java (almost) everything is an object

The Java Programming Language

- Hello World Example.
  - The first line indicates that this is a class whose name is HelloWorld.
  - In java almost EVERYTHING is a class.
  - The next part is the comments part.

    This part is used to document various parts of the code.

    It is good practice to add them, as they improve the readability of your code.

    It is also used by the javadoc program to generate full documentation for your program.

    Java supports C,C++, and Javadoc style comments.

    You should always comment your program. A ratio of 1:1 is reasonable.
- Java Supports all the common flow of control features: if, else if, else.

Flow of Control

```
// If statement:
if (condition){
    // executed if true
}

// if else statement
if (condition){
  // executed if true
} else {
  // executed if false
}

// if else-if statement
if (condition #1){
  // executed if condition #1 is true
} else if (condition #2){
  // executed if condition #2 is false
}
```

- Like any programming language, java has limited precision.
  - When comparing doubles/floats requires a threshold.
- A good testing methodology will test every path of the code.
- Characters are compared using the unicode number for the character
- Java will short-circuit a condition. It will stop evaluating a condition if if can figure the answer.
- Java will compare objects based on location in memory. More later.

Loops

```
for (initialization; termination; increment){
  // execute
}

while (condition){
  // execute something
}

do {
  // execute something
} while (condition)

for (initialization; termination; increment){
  // execute
  If (condition){ continue; }
  // execute
}

for (initialization; termination; increment){
  // execute
  If (condition){ continue; }
  // execute
}
```

- Convenient iteration of containers
- Beware of infinite loops
- Sometimes, we may want to implement a loop that specifies no exit condition, eg *while(true){ … }*
- Often, it is easier to test for exit conditions in the middle of a loop, when those conditions may arise
- In such cases, *break* or *continue* may be used to specify the next action

The Math class

- The Math class provides various methods related to mathematical calculations.
- All the methods are static, that is you do not need to instantiate any object to use these methods.
- Here is a list of some very useful methods from the class.
  - Math.pow(double x, double y) x^y
  - Math.sin(double x) x is in radians
  - Math.asin(double x) x is between -1 and 1.
  - Math.abs(double x) |x|
  - Math.abs(float x) |x|
  - Math.exp(double x) e^x
  - Math.log(double x) log(x)
  - Math.nextUp(double x) The double number that is ``next'' to x but greater than x.
  - Math.random() A random number in [0,1)

The Math Class The Naive & Smart

- Suppose you want to calculate: 12x^4 + 7x^3+3x^2+ 2x+2.

```
public static method(double x){
  return 12 * Math.pow(x,4) + 7 * Math.pow(x,3) + 3 * Math.pow(x,2) + 2*x +2;
}

public static method(double x, double[] coeff){
  tmp =x;
  sum = coeff[0];
  for ( int i = 1; i <coeff.length; ++i){
    sum += coeff[i] * tmp;
    tmp *= x;
  }
  return sum;
}
```

The String Class

- The String Class is a very basic class in Java but it's not a primitive like int, double, float, etc.
- A String object can not be changed! It is *immutable*
- The String class has a mixture of static, general purpose, methods, and class specific methods
- To concatenate Strings, you can use the + operator
- Here are some other useful methods:
  - format(String format, Objects$...$args)\\
  - A general mechanism for formatting output in a C like style. String.format("%-2.3f %7d, 2.34313, 3);\\
  - valueOf(double d) The string representation of the number.
  - compareTo(String s) compare the string to s lexicographically.
  - split(String regex) split the string along the regex.
  - length() & The length of the string.
  - trim() & cut white spaces on both sides.

Java General Concepts

- Java is an object oriented programming (OOP) language
- It revolves around classes and objects
- OOP tries to solve real life problems by modeling the problems as it appears in real life
- Every element in the world – me, you, stock, money, to name a few – has a state and a behavior that may change that state
- The state of a bond could be, for example, time and yield to maturity
- The behavior of the bond could be change the yield to maturity, pay coupon, etc.
- Me. My state could be where I stand, what I point to.
- My behavior could be to walk, to raise my hand, etc.
- As you can see, almost any physical system can be modeled as an object

Java General Concepts

- In an object oriented language there are classes and there are objects
- A class is a template for modeling the behavior of an object.
- An object is a specific instance of the class, with a specific state.
- For example, consider the bond from the previous slide. Suppose it matures in 3 years and has a yield to maturity of 20%. That bond is an instance of the Bond class
- A class has fields and methods.
- A field is a variable that holds a specific state.
- A method can be used to modify fields
- The first type of field is instance specific – it is not shared among objects
- The static field is shared by all objects of the same class

Java General Concepts

- Methods can take as many inputs as necessary. They operate on the input and the state, and can cause a state change. In addition they can return a result to the calling function.
- Many big problems can be broken into small and simple problems.
- Classes and methods should represents these incremental steps toward the solution of the big problem.

Java memory model

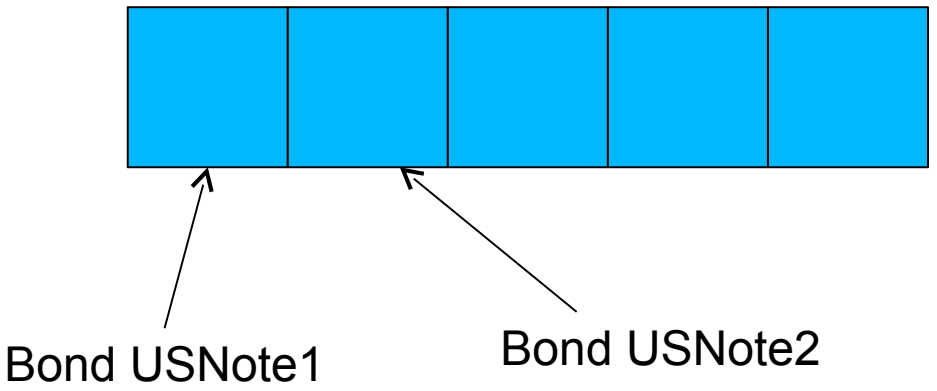
- Three areas of memory: Stack, heap, and “other”
- “Other” memory is used for storing the JVM and related programs
- Stack is where all named variables are stored
- Heap is where all objects and arrays are stored
- Java provides a significant number of tools for monitoring and profiling the memory usage

JVM memory model

- For primitives – int, float, double, etc. – the stack contains the actual contents of the variable – the bytes representing its value
- For non-primitives – such as arrays, Strings, and all Java objects – the stack contains the address of a space on the heap that contains the bytes representing the value of the variable
- When a method is called, stack space for its variables is created
- When execution exits that method, its stack space is de-allocated
- Heap space is de-allocated when no variable holds its address

Java Memory Model: The Heap

- The heap is an area in the memory that is used for storing objects
- Every object, upon creation is allocated space in the heap
- Java tries to keep all the objects as condense as possible.



Java Memory Model: Pass by Value

- Java manipulates all non primitive types as references
- Java passes method arguments by value
  - When Java needs to pass an object to a method it generates a new and independent reference to the object on the method's stack and copies the address of the variable into that reference
  - This can cause a lot of confusion and errors.

Arrays

- Many times we need to hold a collection of similar elements in some order
- For example, we would like to hold the closing prices of a security as a time series
- An array is one data structure that we use for creating a list of elements, but Java offers others
- Arrays are not resizable.
- Java provides a utility for efficiently manipulating arrays
- Arrays.sort(T[] T) for sorting an array.
- Arrays.copyOf(...), Arrays.copyOfRange,... for copying and resizing arrays
- As in C++ or C, array indexing starts at zero (not 1)
- Use array.length to determine the length of the array

Packages, Classes, and beyond

- While java provides many built in and useful classes, most of these are generic and you must still know how to write your own.
- Every class should look as follows:

```
<modifiers> class <name>{
  <state variables>
  <constructors>
  <methods>
}
```

- The modifier(s) can be used for various purposes and there can be more than one (to be discussed)
- *public*, *private*, *protected*, and (empty) can be used for access control
- *static* can be used to define the scope of the class.
- *final* is another possible modifier.

Variables

- Variables are defined as follows:
- <modifiers> <type> name (= default value)
- The modifier can be one of the following: public, protected, private, (empty) for access control.
- *static* allows a variable to be accessed by all instances of the class.
- *final* is use to make constant variables, variables that cannot be modified
- It is very common to avoid defining the variables as public
- For each variable we will write a getName() and setName(type value) methods that can be used to access the variable
- This is very useful to control access to instance variables, and, becomes even more useful when combined with other OOP features

Methods

- Methods are defined similarly to variables:
- <modifiers> <return type> name (parameters){ ... }
- Again the modifier can take the various modifiers discussed earlier.
- When should a method be static?
- A method that needs to perform a task that does not depends on the state of the object should be static.
- For example, all the methods of the Math class are static.
- Also, when the action is not object specific, like Double.isNaN(double x).

Overloading and constructors

- Java, like other object oriented languages, provides an overloading mechanisms.
- We can define many methods with the same name, but having different arguments.
- Java can decide which method to invoke based on the exact parameter list supplied by the caller.
- Finally, every class needs at least one constructor.
- The constructor is used to initialize the object.
- Via overloading we can define multiple constructors, and based on the signature used by the user, the appropriate constructor will be called.
- In order to avoid duplication, the keyword *this* can be used to call a different constructor from a constructor.

Odds and Ends

- In Java we use == to check for equality of reference (what is sometimes called identity), not equality of value
- If you want to check for equality of value, use .equals instead (and implement it when appropriate.)