Moving from C to Java

The Java Syntax We Know

sequencing	P_1; P_2
blocks (compound statements)	{ P_1 }
conditional (selection)	`if` (*condition*) S_1 `else` S_2
conditional (single selection)	`if` (*condition*) S_1
while loops	`while` (*condition*) S_1
do-while loops	`do` S_1 `while` (*condition*)
for loops	`for` (S_1 ; *condition* ; S_2) S_3

multiple selection	`switch` (*expr*) {
	`case` $const_1$: S_1
	...
	`case` $const_n$: S_n
	`default:` S_{n+1}
	}

Java Types

- Some of the types we know and love are still there
  - Integer: byte, short, char, int, long
  - Floating Point: float, double
- And we even get a boolean type

Java Types

- There are no pointers in Java, but they aren't necessary for referential types.
- All primitive types, i.e. the ones we know about are pass by value.
- All classes, wrappers, interfaces and arrays are actually references
- (Will go back to classes and arrays, leaver wrappers and interfaces for later)

The basics

- Everything in life has 3 important parts (not to say the other parts aren't)
  - A state; The things that define this object from a similar one
  - An interface to the state; The ways we can interact with the state of this object
    - Interfaces can be both public and private
  - A set of Behaviours; The things this object can do
- Java classes have all of these things too

The Skeleton Java Class

- Below is a basic Java Class

```
Public class Name
{
     /* State Goes Here*/

     public Name(arguments){} //The constructor

     /*Methods go here*/

     public static void main(String [] args){}
}
```

- All Java classes must have a constructor
- They do not all have to have a main, only your main (Duhh!!!)

State in Java

- The state of a class in Java are the variables global to the class. Why??
- This is because a class encapsulates an object, meaning it contains everything that is an object.
- State can of course be variables of any type

Variables of Class Type

- All class variables are referential types (think pointers)
- They all have a state, behaviours and a public interface detailing them.
- A giant library of classes in Java can be found in the API (bookmark this) http://java.sun.com/j2se/1.5.0/docs/api/

Java Arrays

- Earlier I said Arrays are referential types, what does this mean??
- In Java when we say `int numbers []` this is similar to `int *numbers`. We create a reference/pointer to a piece in memory but there is nothing there.
- Classes work the same way; `String name` is similar to `char *name`, it is only pointing to memory.

Instantiating And “new”

- Both classes and arrays must be instantiated to be used.
- We do this with the command new, which calls the constructor for us
- String name = new String (“Jeff”)
- In Java we have method overloading, which means we can have many functions with the same name but different arguments, like constructors
- char letters [] = {'a','j','e','f','f','z'};
- String name = new String(letters,1,4);

Interface in Java

- When we say public java class, we are defining interface.
- Here are two main types of interfacing Java allows (more will be explained later)
  - Private; This means it is visible only within the class. State is usually private.
  - Public; This means that when you see the class you see this too. Constructors are public.

Behaviours in Java

- All the methods of a class are its behaviours.
- In C a function in only a behaviour because we define it to be, in Java a function is only not a behaviour when we define it that way.
- Because methods are native to the class they can access all class objects

Methods Example

- ```
  public class IntHolder
  {
      private int myNum;

      public IntHolder(int i) {myNum=i;}

  //Below is a very basic selector
      public int getNum()
              {
                      return myNum;
              }
  }
  ```
- Then when we want to use this method we simply go
  ```
  IntHolder temp = new Intholder(5);
  temp.getNum()
  ```
  ```

Back To The Basics

- Everything in life has 3 important parts (not to say the other parts aren't)
  - A state; The things that define this object from a similar one
  - An interface to the state; The ways we can interact with the state of this object
    - Interfaces can be both public and private
  - A set of Behaviours; The things this object can do
- Java classes have all of these things too

The SE view on life

- Lets take the example of a Cat
  - State = things like colour, breed, state of hunger, cleanliness, etc
  - Public interface = We can see the colour, we can feed the cat, we can pet the cadet.
  - Private interface = Only the cat can interface with its thoughts, you or I cannot (We don't even really KNOW the cat has thoughts)
  - A set of behaviours = The cat can eat, sleep, preen, terrorize mice, etc

Cat in C

- To define the state we create the type

```
Typedef struct
{
      char* name;
      char* breed;
      int hunger;
}Cat;
```

- The public interface is the contents of the header file
- The private interface would be any internal functions
- To define the behaviour we create other functions

```
void eat(Cat *chat, int foodAmount)
{
      chat->hunger = (chat->hunger-foodAmount)<0?0:(chat->hunger-
                        foodAmount);
}
```

Cat in Java

```
public class Cat
{//The state
    private String name;
    private String breed;
    private int hunger;

    public Cat(){} // This is a basic constructor

//This behaviour
    public void eat(int foodAmount)
    {
          hunger = (hunger-foodAmount)<0?0:(hunger-
                        foodAmount);
    }
}
```

Classes, Modules &SE Design (Oh My)

- The most obvious of the qualities is of course modularity
- Abstraction; By viewing computer objects like real life object we can achieve a higher level of abstraction
- Information Hiding; Through public and private interfaces.