COMP 202 Review

Calem Bendell
McGill Computer Science Undergraduate Society
calem.j.bendell@gmail.com

This Review

Much of this review will be presented in the form of code samples in an editor and run, where we will discuss the reasoning behind the written code. If you're just reading these slides, try to find the accompanying bits of code, which will be posted nearby after the review.

Variables

```
int x = 5;
```

What this really means is that there will be a reference to some place in the computer's memory, and after that reference is made the value will be made to be 5.

```
int x;
x = 5;
```

essentially equivalent

Looping

```
int i = 0;
while(i < 10){
  i++;
  System.out.println(i);
}

for(int i = 0; i < 10; i++){
  System.out.println(i);
}
```

Variables and Types

- In java, every variable has a type. Everything. If something doesn't have a type to it, the code won't compile.
- In order for something to change type, it has to be specified that it's changing type. This is casting.
- You can cast freely from smaller types to bigger, but not the other way around
- This is ”lossy conversion” and makes casting rules easier to think about

Casting Rules

```
int x = 5; // integer has 32 bits

short y = x; // not allowed, short is only 16 bits

byte z = x; // you’re just making it worse man!
//  (byte is 8 bits)

long a = x; // long is 64 bits, this is all good
```

Scope

After the ”braces close” on an item, it generally falls out of scope

```
{
  int x = 5;
}
System.out.println(x); // that ain’t good
```

Casting for Strings and Chars

```
char c = ’c’;
String s = c; // this doesn’t seem lossy at first
// however, it won’t compile
// a string isn’t just a longer character
// it’s an array of characters
// also a string isn’t a primitive
```

Arrays

```
int numberOfElements = 10;
// all elements will be 0
int[] intArray = new int[numberOfElements];

// now the 6th element is 6
intArray[5] = 6;

// initialise array values with variable
int[] arr2 = {4, 5, 1, 3, 9, 8};
```

Methods

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

public static Boolean returnTrueAlways(){
  return true;
}
```

More Complicated Problems

Are the two below lines equivalent?

```
x > 10 && x < 1000 || x == 3000
((x > 10) && (x < 1000)) || (x == 3000)
```