Chapters 1-4 Summary

These slides are brief summary of chapters 1-4 for students already familiar with programming in C or C++.

Syntax - Java or C?

```
int x[]={1,2,3,4,5,6,7,8,9,10};
int i;
int sum=0;
float average;
for(i=0; i<10; i++){
   sum = sum+x[i];
}
average = sum/10;
```

Syntax - Java or C?

```
void main(void){
    int x[]={1,2,3,4,5,6,7,8,9,10};
    int i;
    int sum=0;
    float average;
    for(i=0; i<10; i++){
       sum = sum+x[i];
    }
    average = sum/10;
    printf("%f\n",average);
}
```

Syntax - Java or C?

```
class Average{
   public static void main(String[] args){
        int x[]={1,2,3,4,5,6,7,8,9,10};
        int i;
        int sum=0;
        float average;
        for(i=0; i<10; i++){
           sum = sum+x[i];
        }
        average = sum/10;
      System.out.println(average);
   }
}
```

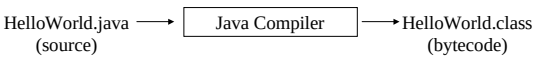
Compiling a program

- Source code - HelloWorld.java
  - viewed with an editor
  - understandable by a human
- Object code - HelloWorld.class
  - for Java, this is machine independent byte code
  - compilers for other languages produce machine code
  - this is also called the binary form or executable

Compiling

- Create HelloWorld.java with an editor
- Execute the command:

```
javac HelloWorld.java
```



Running your Java program

- Once it compiles with no errors, type:
  `java HelloWorld`
- Notice it is not `HelloWorld.class`.
- The name here must be the name found after the keyword `class` in your programs source file. In general it should be the same as the name of the file, minus the extension.

```
// SimpleInput.java-reading numbers from the keyboard
import tio.*;    // use the package tio

class SimpleInput {
  public static void main (String[] args) {
    int width, height, area;

    System.out.println("type two integers for" +
              " the width and height of a box");
    width = Console.in.readInt();
    height = Console.in.readInt();
    area = width * height;
    System.out.print("The area is ");
    System.out.println(area);
  }
}
```

Using tio

- Download tio.jar onto your computer (http://www.cse.ucsc.edu/~charlie/java/tio/)
- Depending upon your OS you can either
  - put tio.jar in JAVA_HOME\jre\lib\ext
  - or modify your CLASSPATH environment variable to include somePath/tio.jar.
- Detailed instructions are at the link above.

Numbers

- standard lengths for byte, short, char, int, long, float, and double
- char is 16 bit unsigned - Unicode not ASCII

Naming Conventions

- SomeClass
- someMethod()
- someVariable
- some_variable
  - preferred by some for private/local variables
- CONSTANT

boolean

- standard type
- has values `true` and `false`
- `if (x = y)`
  - syntax error
- no conversion between int and boolean
- if/while/for all same as C except that the test expression is type boolean

logical operators

- short circuit like C
- && - and
- || - or
- ^ - XOR
- ! - not
- & - non-short circut and
- | - non-short circut or

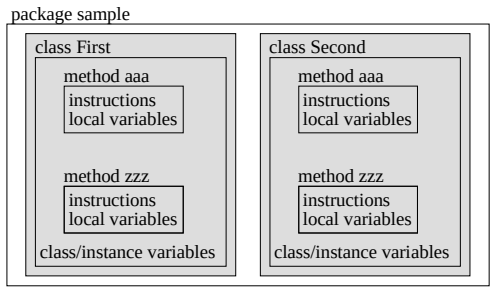
methods = functions

- mimic C functions by adding keyword `static`
- always part of a class
- no & or * for output parameters of primtive types - more about pointers soon

```
// Min2.java: return expression in a method

class Min2 {
  public static void main(String[] args) {
    int   j = 78, k = 3 * 30, m;
    System.out.println(
          "Minimum of two integers Test:");
    m = min(j, k);
    System.out.println("The minimum of : "
       + j + " , " + k + " is  " + m);
  }
  static int min(int a, int b) {
    if (a < b)
      return a;
    else
      return b;
  }
}
```

Java Program Structure



Method Overloading

- 1st new concept
- there can be two methods with the same name, distinguished only by their signatures
  - recall that a method/functions signature is specified by the name plus the parameter types
  - the signature of min() give earlier is `int min(int, int)`
- `System.out.println()` is overloaded to accept a `String` or any of the primitive types.

```
// method overloading
class Min2 {
  public static void main(String[] args) {
    ...
  }
  static int min(int a, int b) {
    if (a < b)
      return a;
    else
      return b;
  }
  static double min(double a, double b) {
    if (a < b)
      return a;
    else
      return b;
  }
}
```

```
//AmbiguousOverload.java: won't compile
class AmbiguousOverload {
  public static void main(String[] args) {
    int i = 1, j = 2;

    System.out.println(ambig(i,j));
  }
  static boolean ambig(float x, int y){
    return x < y;
  }
  static boolean ambig(int x, float y){
    return x < y;
  }
}
// The problem is the method call not the method
// definitions.
```

Scope

- Variable declarations can be introduced at anytime.
- ```
  for(int i = 0; i < x; i++) {
      ...
  }
  // i is undefined here
  ```
- You cannot have overlapping local definitions.

```
public class SquareRoots2 { // contains scope errors
  public static void main(String[] args) {
    int i = 99;
    double squareRoot = Math.sqrt(i);
    System.out.println("the square root of " + i +
                       " is " + squareRoot);
    for (int i = 1; i <= 10; i++) {
      double squareRoot = Math.sqrt(i);
      double square = squareRoot * squareRoot;
      System.out.println("the square root of " + i +
                         " is " + squareRoot);
      System.out.println("squaring that yields " +
                          square);
    }
    System.out.println("The final value of square"
                      + " is " + square);
  }
}
```

Non-primitive Types

- Non-primitive types: classes (and arrays)
- Their values: objects
- Create objects with:
  `new TypeName(...)`
- Except `String` which has syntactic support with `"some string"`.
- Declare variables just like primitive types.

Operations on Objects

- Except for String, there are no built in operators for operating on objects (as there are for numeric types, such as + and * for int).
- Instead, operations are performed on objects using methods.

Example operations on objects

```
Point p = new Point(10, 20);
p.translate(100, 200);
System.out.println(p);
```

Prints:

```
java.awt.Point[x=110,y=220]
```

```
String s = "testing";
int x = s.length();
int y = "even this works".length();
int z = ("this " + "or that").length();
System.out.println(x + ", " + y + ", " + z);
```

Prints:

```
6, 15, 12
```
```