Programming Concepts Using Java

Week 2 Revision

Getting started

- Java program to print hello, world

```
public class HelloWorld{
    public static void main(String[] args) {
        System.out.println("hello, world);
    }
}
```

- A Java program is a collection of classes
- All code in Java lives within a class
- Modifier public specifies visibility
- The signature of main( )
  - Input parameter is an array of strings; command line arguments
  - No output, so return type is void
- Write once, run anywhere

Scalar types

- Java has eight primitive scalar types
  - int, long, short, byte
  - float, double
  - char
  - boolean
- We declare variables before we use them

```
int x, y;
x = 5;
y = 10;
```

- Characters are written with single-quotes (only)

```
char c = 'x';
```

- Boolean constants are true, false

```
boolean b1, b2;
b1 = false;
b2 = true;
```

Scalar types

- Initialize at time of declaration

```
flat pi =  3.1415927f;
```

- Modifier final indicates a constant

```
final float pi = 3.1415927f;
```

Operators

- Arithmetic operators are the usual ones

  `+, -, *, /, %`

- No separate integer division operator //
- When both arguments are integer, / is integer division
- No exponentiation operater, use Math.pow()
- Math.pow(a,n) returns a^n
- Special operators for incrementing and decrementing integers

```
int a = 0, b = 10;
a++; // Same as a = a+1
b--; // Same as b = b-1
```

- Shortcut for updating a variable

```
int a = 0, b = 10;
a += 7; // Same as a = a+7
```

Strings

- String is a built-in class
- String constants enclosed in double quotes

```
String s = "Hello", t = "world";
```

- $+$ is overloaded for string concatenation

```
String s = "Hello";
String t = "world";
String u = s + " " + t;
// "Hello world"
```

- Strings are not arrays of characters
- Instead use s.charAt(0), s.substring(0,3)

Arrays

- Arrays are also objects
- Typical declaration

```
int[] a;
a = new int[100];
```

- Or int a[] instead of int[] a
- a.length gives size of a
- Array indices run from 0 to a.length-1

Control flow

- Conditional execution

```
if (condition) { ... } else { ... }
```

- Conditional loops

```
while (condition) { ... }
do { ... } while (condition)
```

- Iteration - Two kinds of for
- Multiway branching – switch

Classes and objects

- A class is a template for an encapsulated type
- An object is an instance of a class

```
public class Date {
    private int day, month, year;
    public Date(int d, int m, int y){
        day = d;
        month = m;
        year = y;
    }
    public int getDay(){
        return(day);
    }
}
```

- Instance variables - Each concrete object of type Date will have local copies of date, month, year

Creating and initializing objects

- new creates a new object
- How do we set the instance variables?
- Constructors — special functions called when an object is created
  - Function with the same name as the class
  - d = new Date(13,8,2015);
- Constructor overloading - same name, different signatures
- A constructor can call another one using this
- If no constructor is defined, Java provides a default constructor with empty arguments
  - new Date() would implicitly invoke this
  - Sets instance variables to sensible defaults
  - For instance, int variables set to 0
  - Only valid if no constructor is defined
  - Otherwise need an explicit constructor without arguments

Copy constructors

- Create a new object from an existing one

```
public class Date {
    private int day, month, year;
    public Date(int d, int m, int y){
        day = d; month = m; year = y;
    }
    public Date(Date d){
        this.day = d.day; this.month = d.month; this.year = d.year;
    }
}
public class UseDate() {
    public static void main(String[] args){
        Date d1,d2;
        d1 = new Date(12,4,1954); d2 = new.Date(d1);
    }
}
```

Basic input and output in java

- Reading input
  - Use Console class
  - Use Scanner class

```
Scanner in = new Scanner(System.in);
String name = in.nextLine();
int age = in.nextInt();
```