CS206 Intro to Data Structures

Spring 2020

Course Goals

1. Become a better computer scientist
2. Learn about common data structures
   1. Implementation
   2. How and when to use each
3. Understand Object Oriented program design and its implementation in Java
4. Develop an understanding of UNIX
5. Become a better Java programmer

Things to Know

- Course website
  - www.cs.brynmawr.edu/cs206
    - usually updated after each class
- Syllabus
  - www.cs.brynmawr.edu/cs206/syllabus.html
    - usually updated on weekend for next weeks material
- Homeworks
  - Approximately weekly.
  - Typically due on Thursday before midnight
  - Help in lab (Park 231) Sunday-Thursday 6:30-9:30 pm
    - starting next week

More Things to Know

- CS account
  - If you do not have a cs account, you will
- Lab:
  - TH 2:25pm-3:45pm
  - Attendance is required (and part of grade)
  - There is lab this week
- Software: Java, Visual Studio Code, Unix

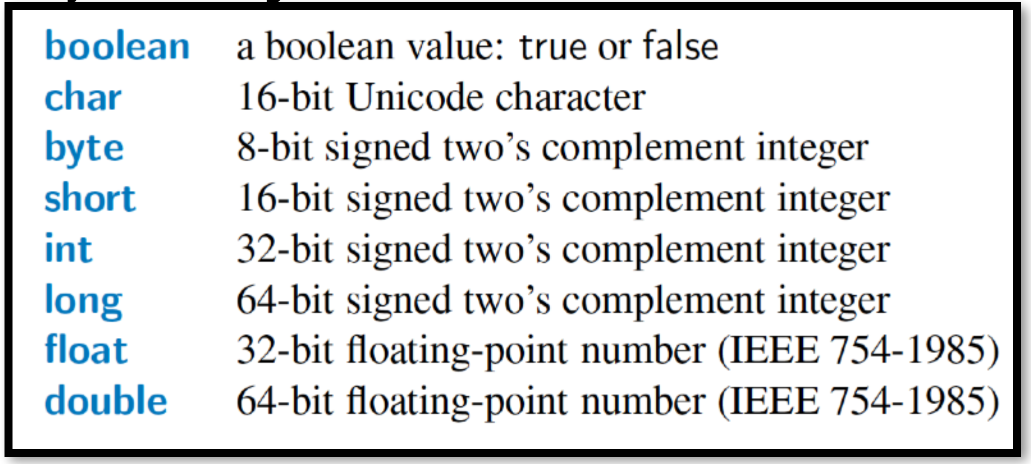
Java

- “Object Oriented” Language
- Data Types
  - Base
    - fixed set
    - Initial lower case letter (e.g. int)
  - Objects (Classes)
    - User extensible
    - Initial capital letter (by convention)

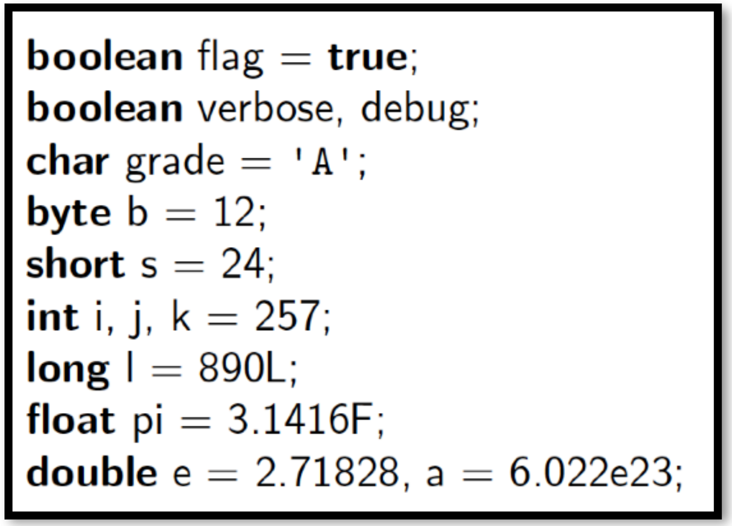
Base/Primitive Types

- Primitive types define memory used to store the data

Extant definitions of primitives subject to change



boolean	a boolean value: true or false
char	16-bit Unicode character
byte	8-bit signed two's complement integer
short	16-bit signed two's complement integer
int	32-bit signed two's complement integer
long	64-bit signed two's complement integer
float	32-bit floating-point number (IEEE 754-1985)
double	64-bit floating-point number (IEEE 754-1985)



```
boolean flag = true;
boolean verbose, debug;
char grade = 'A';
byte b = 12;
short s = 24;
int i, j, k = 257;
long l = 890L;
float pi = 3.1416F;
double e = 2.71828, a = 6.022e23;
```

Classes and Variables

- A class is a description of what an object stores (its data) and how it functions
  - ▫ instance variables
  - ▫ methods
- ▫ Every variable is either a base type or a reference to an object
- Every object is an instance of a class

Creating and Using Objects

- In Java, a new object is created by using the `new` operator followed by a call to a constructor for the desired class.
- A constructor is a special method that shares the same name as its class. The new operator returns a reference to the newly created instance.
  - every method other than a construction must give the type of information it returns
- **Almost everything in Java is a class**
  - More properly, almost all variables in Java store references to instances of a class

Class Example

```
public class InchWorm
{
    private int measurement;
    public InchWorm() {
        this.measurement=1;
    }
    public InchWorm(int startingMeasurement) {
        this.measurement = startingMeasurement;
    }
    public InchWorm(InchWorm iw) {
        this.measurement = iw.getMeasurement();
    }
    public int getMeasurement() {
        return this.measurement;
    }
    public void doubleM() {
        this.measurement *= 2;
    }
    public String toString() {
        return "The marigold measures " + this.measurement + " inc
    }
    public void reset() {
        this.measurement=1;
    }
```

Class Part2

```
    public static void main(String[] args) {
        InchWorm inchworm = new InchWorm();
        inchworm.doubleM();
        System.out.println(inchworm);
        InchWorm inchworm2 = new InchWorm(inchworm);
        inchworm2.doubleM();
        System.out.println(inchworm2 + " " + inchworm);
     }
}
```

Access Control Modifiers

- `public` — all classes may access
- `private` — access only within that class.
- "" (read as package) — access only by classes within the package
    - (I hate significant whitespace)
  - The package is generally the code you are working on.

Static

- When a variable or method of a class is declared as `static`, it is associated with the class as a whole, rather than with each individual instance of that class.
- Only acceptable use (at least for this course):
  - In methods …
    - `public static void main(String[] args)`
  - In variables .. to declare constants
    - `public static final double GOLDEN_MEAN =1.61803398875;`

Casting (of base types)

- Assignment REQUIRES type equality
- Use casting to change type
- Must explicitly cast if there is a possible loss of precision

```
private void trial()
    {
        int x = 5;
        double y = 1.2;
        y = x;
        ~~x = y;~~

        y = (double) x;
        x = (int) y;
    }
```

Object Casting

- Widening cast –
  - to something that was extended from
- Narrowing cast –
  - to an extended class
- Java will perform an implicit widening cast, but not a narrowing
  - Narrowing cast may assume information that is not present.

```
public class Caster {
    private class A {}
    private class B extends A {
        private int bvar;
        public B() { bvar = 1; }
    }
    public void tester() {
        A a = new A();
        B b = new B();
        A aa = b;
        B bb = (B)a;
    }
}
```

`.equals:` Object Equality

- Do not use `==`
  - Use == only when comparing base types
- Review your strings and `String` class methods

```
public class StringEqual {
  public static void main(String[] args) {
    String str1 = new String("one");
    String str2 = new String("one");
    System.out.println("str1==str2: "
             + str1 == str2);
    System.out.println("str1==str2: "
             + (str1 == str2));
    System.out.println("str1.equals(str2): "
             + str1.equals(str2));
  }
}
```

Wrapper Types

- Most data structures and algorithms in Java’s libraries only work with object types (not base types).
- To get around this obstacle, Java defines a wrapper class for each base type.
- Implicitly converting between base types and their wrapper types is known as automatic boxing and unboxing.

Autoboxing and unboxing

```
public class Wrapper
{
    public void w1(Integer ii) {
        System.out.println(ii);
        int i3 = ii; // auto unboxing
        System.out.println(i3*i3);
        System.out.println(i3*ii); // auto unboxing
    }
    public static void main(String[] args) {
        Wrapper w = new Wrapper();
        w.w1(5); // autoboxing
    }
}
```

What you should know/review

- variables
- expressions
- operators
- methods
  - parameters
  - return value
- conditionals
- `for`/`while` loops

- class design and object construction
  - instance variables
  - constructor
  - getters/setters
  - class methods
  - `new`
- arrays
- arrays of objects
- `String`