CS102

Introduction to
data structures, algorithms,
and object-oriented
programming

Writing a Java Application

STEP 1:

- Create a file containing one or more *class* definitions.
  - main body of program starts "public class ClassName", where
  - all code is written within class body, delimited by braces.
  - the prefix of the file name must be the same as class name inside the file.

NOTE:

- Several classes may combine to form a complete program (called an *application*) but at least one class per application must contain the *main* method. Execution always starts at the main method of whatever class is being run.

Writing a Java Application

STEP 2:

- Compile the program in whatever IDE or operating system you are using:
  - translates code into *Java Byte Code*
  - byte code is stored in file with suffix ".class"

STEP 3:

- Run the code in whatever IDE or operating system you are using.
  - Never need to type ".class" suffix, but that is really what is used.

Java features

Java is:

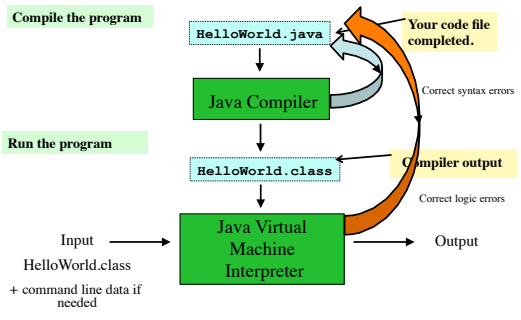
- class-based
- object-oriented
- platform independent (portable)
- concurrent via multi-threading
- both compiled and interpreted

Scheme features

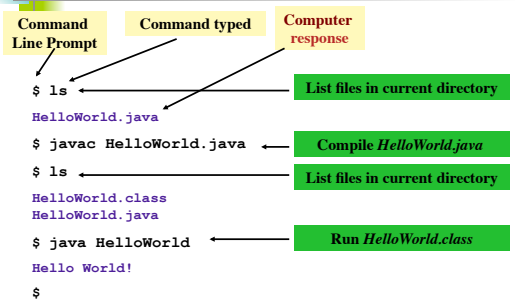
Scheme is:

- function-based
- object-oriented in a limited way
- optimized for recursion
- interpreted line by line for speedier execution
- simple syntax based on algebraic formulas

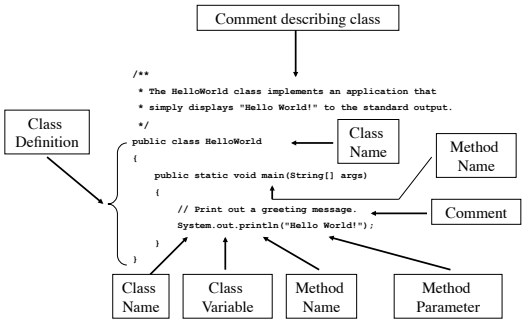
Java coding cycle



Compile and run a Java application file at command line $



HelloWorld.java



HelloWorld.java

```
/**
 * HelloWorld with main method that calls static method.
 */
public class HelloWorld
{
    public static void main(String[] args)
    {
        greeting(); // call to greeting method
    }
    public static void greeting() {
        // Print out a greeting message.
        System.out.println("Hello World!");
    }
}
```

Scheme/Racket hello-world

```
// Thee hello-world class displays "Hello
// World!" to the standard output.

(define (hello-world)
    (printf "Hello world!~%")
 )
```

Java Programs

- Classes are intended to be definitions for data types, sort of like structs in Racket.

The 3 biggest obstacles to new Java programmers:

1. Much of basic code uses high-level concepts
2. Input and output syntax is confusing and verbose
3. Graphics programs are unintuitive

Java Programs

Like Scheme files, libraries of code can be accessed for different tasks in Java.

Java uses a class inheritance mechanism to allow code written in ancestor classes to be used and refined in descendant classes.

NetBeans IDE

DrRacket or DrScheme were IDEs you used to create and run your programs in 101.

There are many different IDEs for Java. We will use the NetBeans and DrJave IDEs. These IDEs are free to download and come packaged with the necessary Java Developer's Kit and the JVM.

If you don't have a CS account, see Jerry, our system administrator (SP307) and let your professor know.

Styles of Java Programs

A Java program (i.e., class) is either:

1. a library of *static* methods (functions) that may return values or just have side effects (like Scheme); or
2. a data type definition used for creation of objects; or
3. both

There is one static method that *must* be included in every set of Java programs: the **main** method.

Each program starts execution at a method with the following signature:

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

Comparison of Racket and Java

Write function to compute interest in Scheme and show same method in Java. DONE

Write function that consumes an integer score and calculates a grade in Scheme and show same method in Java. NOT DONE

Write function to calculate the factorial and show same method in Java. NOT DONE

One difference is that Java functions (called methods) don't run on their own. They must be defined in a class.

Scheme/Racket program

```
(define interest-calculator
  (lambda ()
    (local
      ((define principal 17000)
       (define rate 0.027)
       (define interest 0))
      (begin
        (set! interest (* principal rate))
        (set! principal (+ principal interest)
        (printf "The interest earned is $~a\n"
         interest)
        (printf "The value of the investment
         after one year is $~a\n" principal)
        ))))
```

Java program

```
public class InterestCalculator {  // class declaration line

    public static void main(String[] args) {  //main method start
        double principal;     // The value of the investment.
        double rate;          // The annual interest rate.
        double interest;      // Interest earned in one year.

        principal = 17000;
        rate = 0.027;

        interest = principal * rate;   // Compute and initialize the interest.

        principal = principal + interest;

        System.out.print("The interest earned is $");
        System.out.println(interest);
        System.out.print("The value of the investment after one year is $");
        System.out.println(principal);

    } // end of main
} // end of class InterestCalculator
```