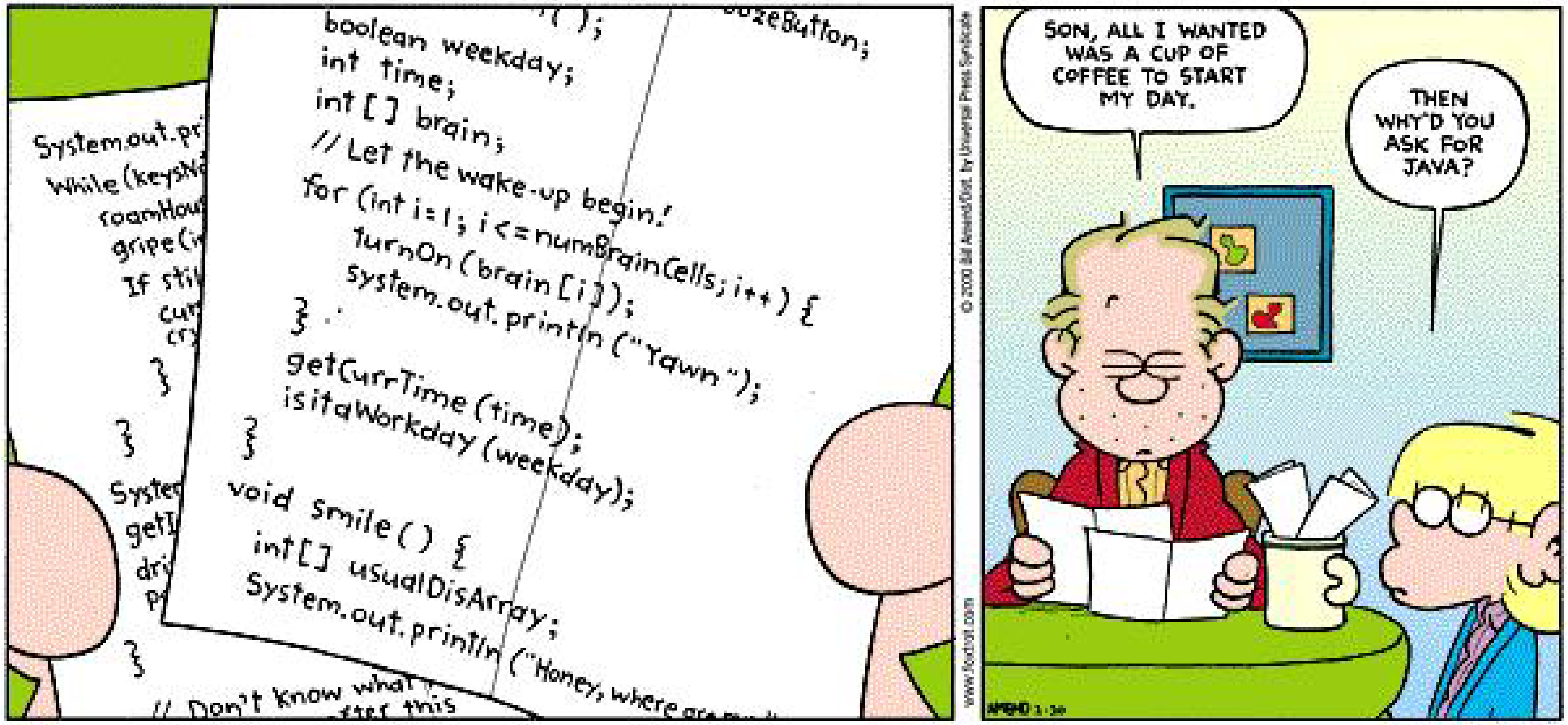
Introduction


SON, ALL I WANTED WAS A CUP OF COFFEE TO START MY DAY.
THEN WHY'D YOU ASK FOR JAVA?
© 2000 Bill Amend/Dist. by Universal Press Syndicate
www.foxtrot.com

```
boolean weekday;
int time;
int [] brain;
// Let the wake-up begin!
for (int i=1; i<=numBrainCells; i++) {
    turnOn (brain[i]);
    system.out.println ("Yawn");
}
getCurrTime (time);
isitaWorkday (weekday);
}
void smile () {
  int[] usualDisArray;
  System.out.println ("Honey, where are my
```

JAVA BASICS

Java Basics

- Classes
  - Most classes are in their own file with a .java extension
  - Name of the class and file are the same
    - Case Sensitive!
  - A *class header* defines the class
  - The *class body* is surrounded by curly brackets

Java Class

```java
public class SimpleClass {

  public static void main(String[] args) {
     System.out.println("Hello world");
     printMore();
  }
  public static void printMore() {
     System.out.println("Hello again");
  }

}
```

Java Class

class header

```
public class SimpleClass {

    public static void main(String[] args) {
        System.out.println("Hello world");
        printMore();

    }
    public static void printMore() {
        System.out.println("Hello again");
    }

}
```

class body

Java Basics (cont.)

- Classes are organized into packages
- Classes contain methods
  - The *method header* defines (declares) the method
    - visibility, return type, name, formal parameters
    - The *method signature* is the name and formal parameter list
  - The *method body* is surrounded by curly brackets
- Methods contain *program statements*
    - Statements end with a semicolon

Java Class

```
public class SimpleClass {

  public static void main(String[] args) {
      System.out.println("Hello world");
      printMore();
  }
  public static void printMore() {
      System.out.println("Hello again");
  }

}
```

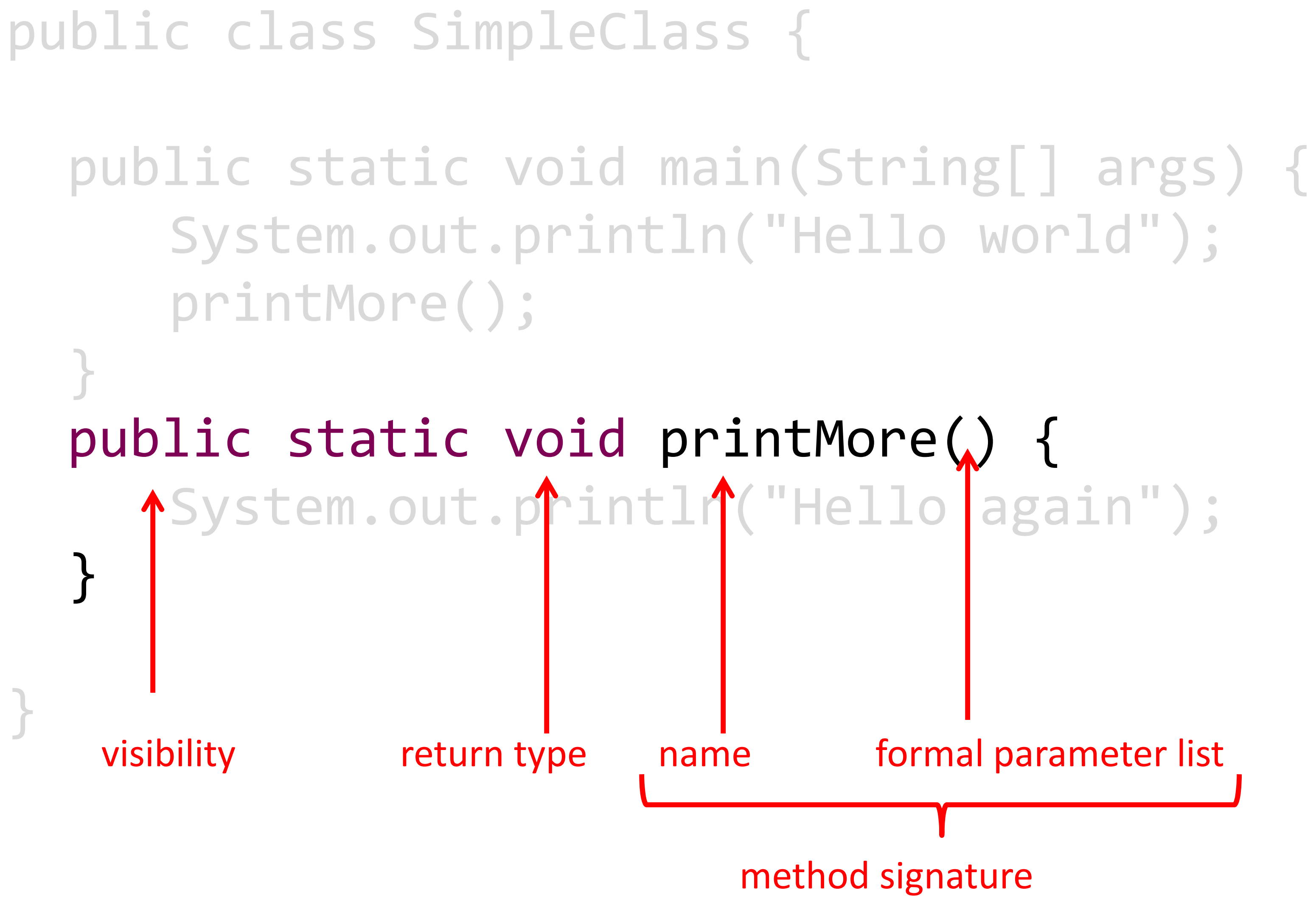
Java Class

```
public class SimpleClass {

  public static void main(String[] args) {
      System.out.println("Hello world");
      printMore();
  }
  public static void printMore() {
      System.out.println("Hello again");
  }

}
```

method header

method body

Java Class



```java
public class SimpleClass {

  public static void main(String[] args) {
     System.out.println("Hello world");
     printMore();
  }
  public static void printMore() {
     System.out.println("Hello again");
  }

}
```

Java Class

```
public class SimpleClass {

   public static void main(String[] args) {
       System.out.println("Hello world");
       printMore();
   }
   public static void printMore() {
       System.out.println("Hello again");
   }

}
```

program statement

The main Method

- main is a special method
  - Launched when you run a program
  - Has this **exact** header

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

Syntax Rules

- Java is case sensitive
- Java ignores whitespace
  - But humans don't!
- Java has two ways to comment code:

  `//` can be used for one-line comments

  `/*` and `*/` can be used to surround comments that span multiple lines

Documenting Code

- You must add documentation to explain your code.
- Few people *enjoy* writing comments, but it is important to write them as you are programming so that your code can be maintained later

“Documentation is the castor oil of programming. Managers know it must be good because the programmers hate it so much.” -Anonymous

Java Naming Conventions

- Classes
  - Single nouns
  - Capital camel case
  - Examples: Student, String, Employee, JFrame
- Methods
  - Lower camel case
  - Descriptive (often a verb)
  - Examples: getID, print, display, calculateTotal

JAVA KEY PRINCIPLES

Object-Oriented Programming

- An *object-oriented* approach lends itself to breaking a problem down into pieces.
- An *object* is a fundamental entity in a Java program.
- Objects can be used to represent real-world entities.
  - Example: An *Employee* object might handle the processing of data for that employee.
  - Example: A *Company* object might manage all employees and the data for the company.
- In Java, all variables are either objects or primitives.

Objects

- An object has:
  - *State*- descriptive characteristics
  - *Behaviors*- things it can do or can be done to it
- The behavior of an object might change its state.
- Example: a *BankAccount* object has:
  - State
    - Account Number
    - Balance
  - Behaviors
    - Make deposit
    - Make Withdrawal

Classes and Objects

- An object is defined by a *class*.
- The class is the *blueprint* of an object.
  - The *BankAccount* class defines what a bank account object will look like.
  - We then create *objects* from that class (e.g., Jess's Account, Jim's Account, etc.)
- *Methods* to define the behaviors of the object
- *Instance data variables* to define the state of the object
- A class represents a concept. An object represents the embodiment of that concept.

Other Key Principles

- Encapsulation
  - Objects protect and manage their own information
  - Objects are self-governing
  - Objects keep their internal state private
  - Example: You can only change the account balanced through the *withdrawal* and *deposit* methods, not directly.
- Inheritance
  - Classes organized into hierarchies (using “is a” relationships)
  - Example: *BankAccount* defines general properties. *SavingsAccount* inherits from *BankAccount* (a savings account is a bank account) but adds additional information about earning interest
- Polymorphism
- These are all “use now, understand later” ideas… ☺

PROGRAMMING PROCESS

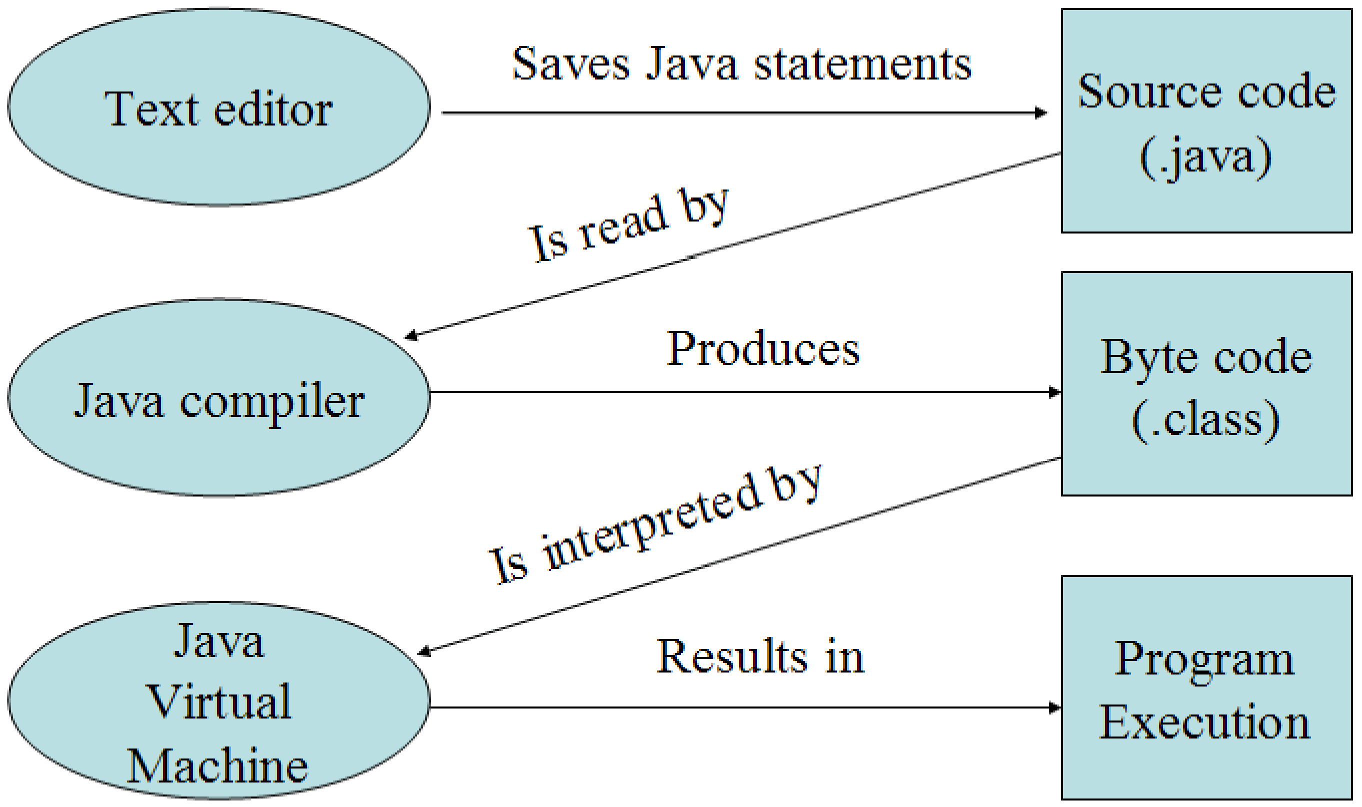
Program Development

- Understand the problem and client specifications
- Design a solution
- Implement the solution
  - Code!
- Test the solution
  - And then back to…

Programming Process

- You write code in a development environment (such as Eclipse)
- The Java compiler translates your code into *bytecode*
  - This is stored in the .class file
- The Java Virtual Machine (JVM) translates *bytecode* into *executable* code
  - Each platform as its own JVM
  - Executable code is platform-dependent
- Java can run anywhere that has JVM
  - This is what makes Java *platform-independent*

Programming Process (cont.)



Graphic From: http://www.cwu.edu/~gellenbe/110/lectures/lecture2.php

Syntax and Semantics

- Syntax
  - Defines how symbols, reserved words, and identifiers can be put together to make a valid program
- Semantics
  - Defines what the statements mean (the purpose of the program)
- A program that is syntactically correct is not necessarily logically (semantically) correct
  - A program will always do what we tell it to do, not what we *meant* to tell it to do!

Errors

- Compile-Time Errors
  - Problems with syntax
  - No bytecode is created
  - Example: missing semicolon, missing bracket
- Run-Time
  - Problems during execution that cause the program to crash
  - Compiles and might run fine under some conditions
  - Example: user enters numbers to be divided and tries to divide by zero
- Logical Errors
  - Problems during execution that produces incorrect results
  - Compiles and might run fine under some conditions
  - Example: using an incorrect formula to calculate temperature

Program Structures

- Text-Based
  - Standalone Application
    - Input from and output to the console
    - Has a `main` method
- Graphic
  - Standalone GUI Application
    - Launch from the console or create a runnable JAR
    - Has a `main` method
  - Applet
    - Embedded in a website
    - No `main` method

TEXT-BASED PROGRAMS

Console Output

- The `System.out` object represents the screen where we can send output
  - The `print` method prints a `String` *without* advancing to the next line
  - The `println` method prints a `String` and advances to the next line

Input from the Console

- The `Scanner` class provides convenient methods to reading input values of various types.
- A `Scanner` object can be set up to read input from various sources, including a user typing values on the keyboard.
  - Keyboard input is represented by the `System.in` object.

The `Scanner` Class

- The Scanner class is part of the `java.util` class library. You must import this library to use the class.

```
    import java.util.Scanner;
```

- The following line creates a `Scanner` object that reads from the keyboard:

```
Scanner scan = new Scanner (System.in);
```

  - The `new` operator creates a `Scanner` object called `scan`

The `Scanner` Class (cont.)

- Once a `Scanner` object is created, it can be used to invoke various input methods.
- The `nextLine` method reads all of the input until the end of the line (carriage return key) and returns it as a string literal:

```
    String answer = scan.nextLine();
```

- Many other methods:
  - google: Java Scanner API
  - http://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html

Java and Eclipse

- Java program organization
  - Packages contain classes
  - Classes contain methods
  - Methods contain statements
- Eclipse
  - Workspace
    - Primary directory for all your work
    - Example: C:/JessicaFiles/CS111B/EclipseWorkspace
  - Projects
    - Used to organize your work
    - Can contain multiple packages
    - Examples: Homework1, Chapter2Class, Project1

Practice

- Write a Hello World program.
- Complete the Java Error Predictions exercise.