Welcome to Advance Java Programming

By

Jitender Singh

Asst. Professor, GITAM

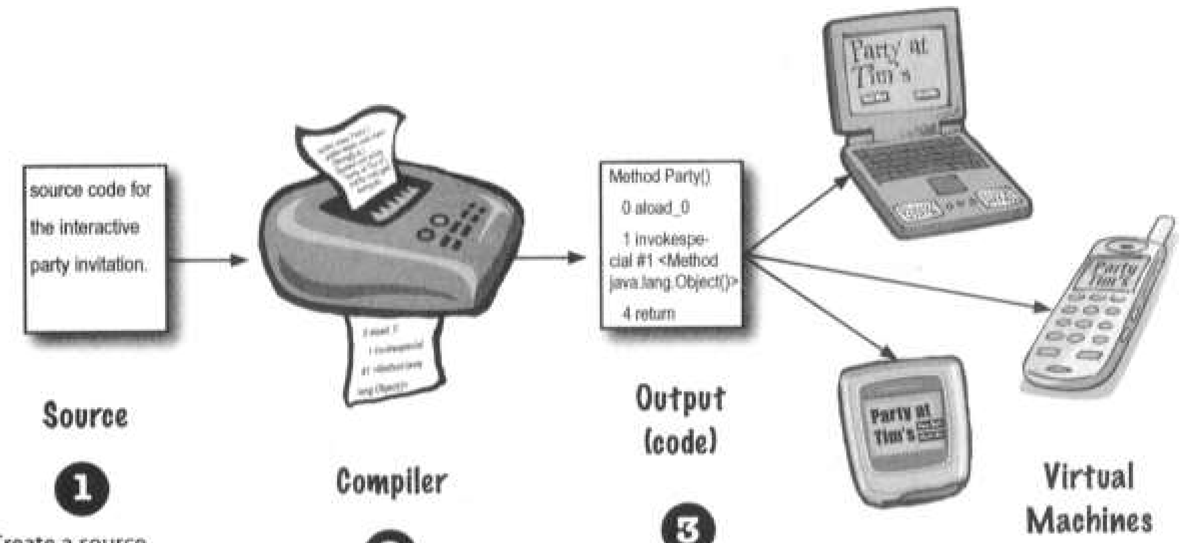
What is Programming Language?

- A programming language is a high level language that contains instructions that controls a computer’s operations.
- Examples: Java, C++, C, Visual Basic, …
- Compiling. A programming language needs to be translated into a low level machine code before execution on a computer.
- Developed by Sun Microsystems (James Gosling)
- A general-purpose Object-Oriented language
- Based on C/C++
- Designed for easy Web/Internet applications
- Widespread acceptance

Features of Java Programming Language

- Simple
- Object-oriented. Object vs. procedure
- Platform Independent
- Safe. No pointers. Live in virtual machine.
- Multi-threaded
- Garbage collected
- Encapsulation
- Inheritance
- Polymorphism

How Java Works --1


source code for the interactive party invitation.
Source
1
Compiler
2
Method Party()
0 aload_0
1 invokespecial #1 <Method java.lang.Object()>
4 return
Output (code)
3
Party at Tim's
Virtual Machines
4

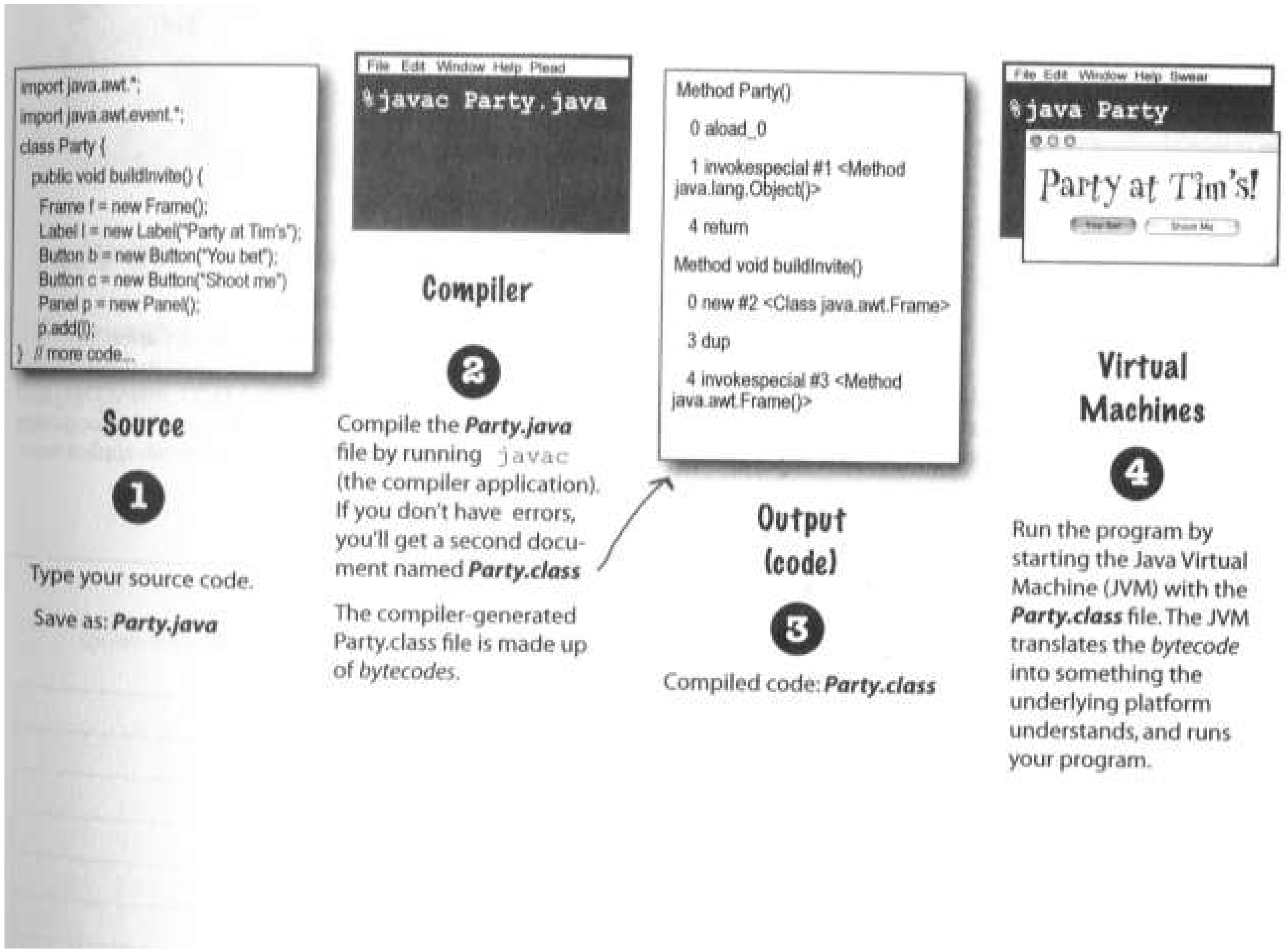
Create a source document. Use an established protocol (in this case, the Java language).

Run your document through a source code compiler. The compiler checks for errors and won't let you compile until it's satisfied that everything will run correctly.

The compiler creates a new document, coded into Java ***bytecode***. Any device capable of running Java will be able to interpret/translate this file into something it can run. The compiled bytecode is platform-independent.

Your friends don't have a physical Java Machine, but they all have a ***virtual*** Java machine (implemented in software) running inside their electronic gadgets. The virtual machine reads and *runs* the bytecode.

How Java Works –2



```
import java.awt.*;
import java.awt.event.*;
class Party {
  public void buildInvite() {
    Frame f = new Frame();
    Label l = new Label("Party at Tim's");
    Button b = new Button("You bet");
    Button c = new Button("Shoot me")
    Panel p = new Panel();
    p.add(l);
} // more code...
```

Source

1

Type your source code.

Save as: ***Party.java***

```
%javac Party.java
```

Compiler

2

Compile the ***Party.java*** file by running `javac` (the compiler application). If you don't have errors, you'll get a second document named ***Party.class***

The compiler-generated Party.class file is made up of *bytecodes.*

```
Method Party()
  0 aload_0
  1 invokespecial #1 <Method
java.lang.Object()>
  4 return
Method void buildInvite()
  0 new #2 <Class java.awt.Frame>
  3 dup
  4 invokespecial #3 <Method
java.awt.Frame()>
```

Output (code)

3

Compiled code: ***Party.class***

```
%java Party
```

Virtual Machines

4

Run the program by starting the Java Virtual Machine (JVM) with the ***Party.class*** file. The JVM translates the *bytecode* into something the underlying platform understands, and runs your program.

Three most used Java command and Platform Editions

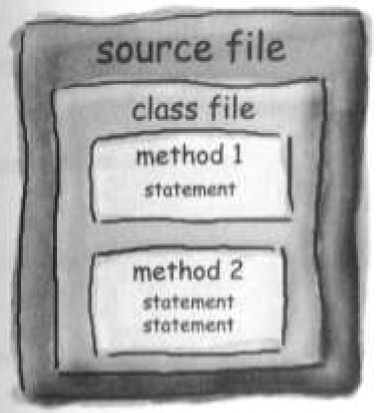
- javac -- Compile java source code into Byte code.
- java -- Run a Java application.
- jar -- Archive files.
- There are 3 Java Platform Editions
- Java 2 Platform, Standard Edition (J2SE)
- Core Java Platform targeting applications running on workstations
- Java 2 Platform, Enterprise Edition (J2EE)
- Component-based approach to developing distributed, multi-tier enterprise applications
- Java 2 Platform, Micro Edition (J2ME)

  Targeted at small, stand-alone or connectable consumer and embedded devices

Brief History of Java

- In 1990, Sun Microsystems began an internal project known as the Green Project to work on a new technology.
- In 1992, the Green Project was spun off and its interest directed toward building highly interactive devices for the cable TV industry. This failed to materialize.
- In 1994, the focus of the original team was re-targeted, this time to the use of Internet technology. A small web browser called HotJava was written.
- Oak was renamed to Java after learning that Oak had already been trademarked.
- In 1995, Java was first publicly released. • In 1996, Java Development Kit (JDK) 1.0 was released. • In 2002, JDK 1.4 (codename Merlin) was released, the most widely used version. • In 2004, JDK 5.0 (codename Tiger) was released, the latest version.

Java Code Structure --1

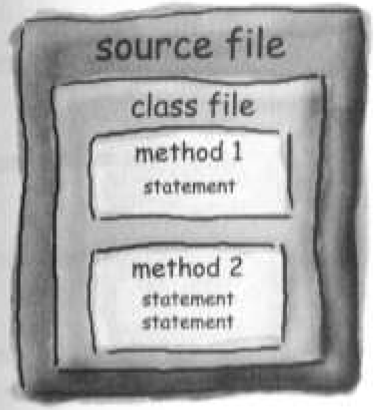


Put a class in a source file.

Put methods in a class.

Put statements in a method.

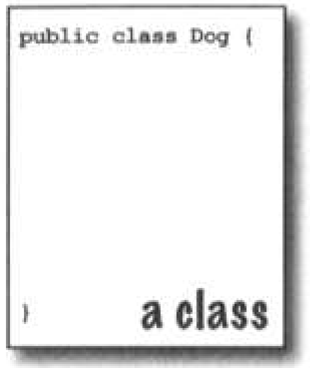
Java Code Structure--2



Put a class in a source file.

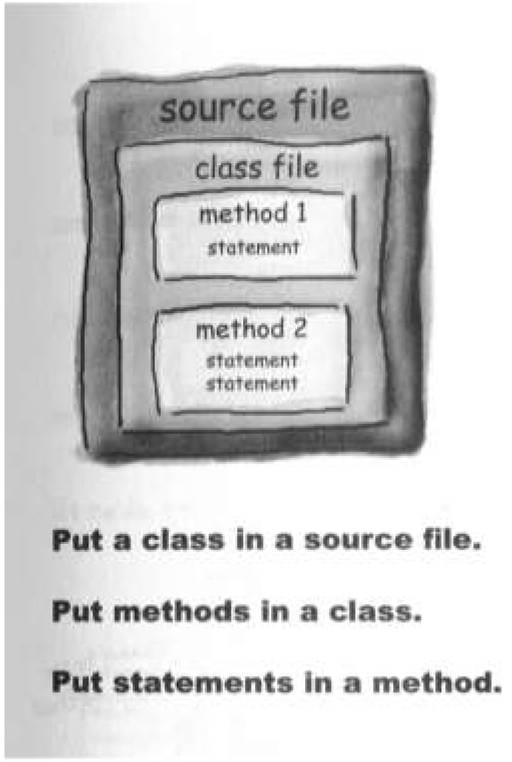
Put methods in a class.

Put statements in a method.



```
public class Dog {

}
```

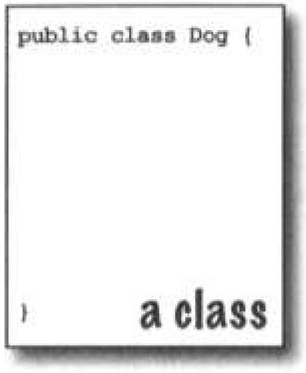
Java Code Structure --3



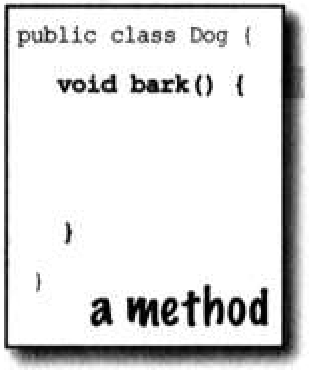
Put a class in a source file.

Put methods in a class.

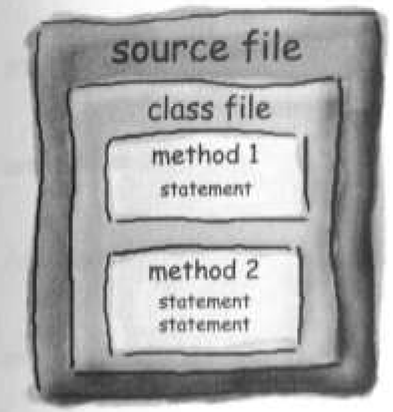
Put statements in a method.



```
public class Dog {

}
```



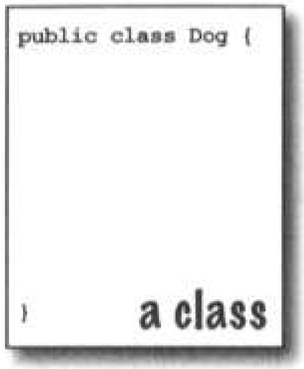
```
public class Dog {
   void bark() {

   }
}
```



Put a class in a source file.

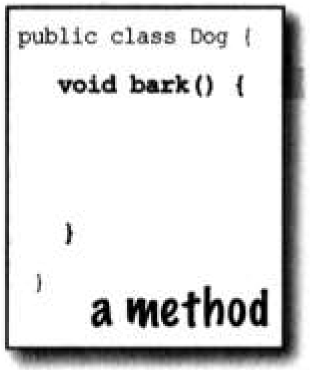
Put methods in a class.

Put statements in a method.



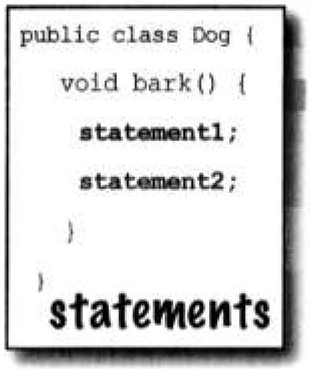
```
public class Dog {

}
```

a class



```
public class Dog {
   void bark() {

   }
}
```

a method



```
public class Dog {
   void bark() {
     statement1;
     statement2;
   }
}
```

statements

Your First Cup of Java --1

```
public class HelloWorld
{
   public static void main(String[] args)
   {
    System.out.println(“Hello World!");
   }
}
```

Your First Cup of Java --2

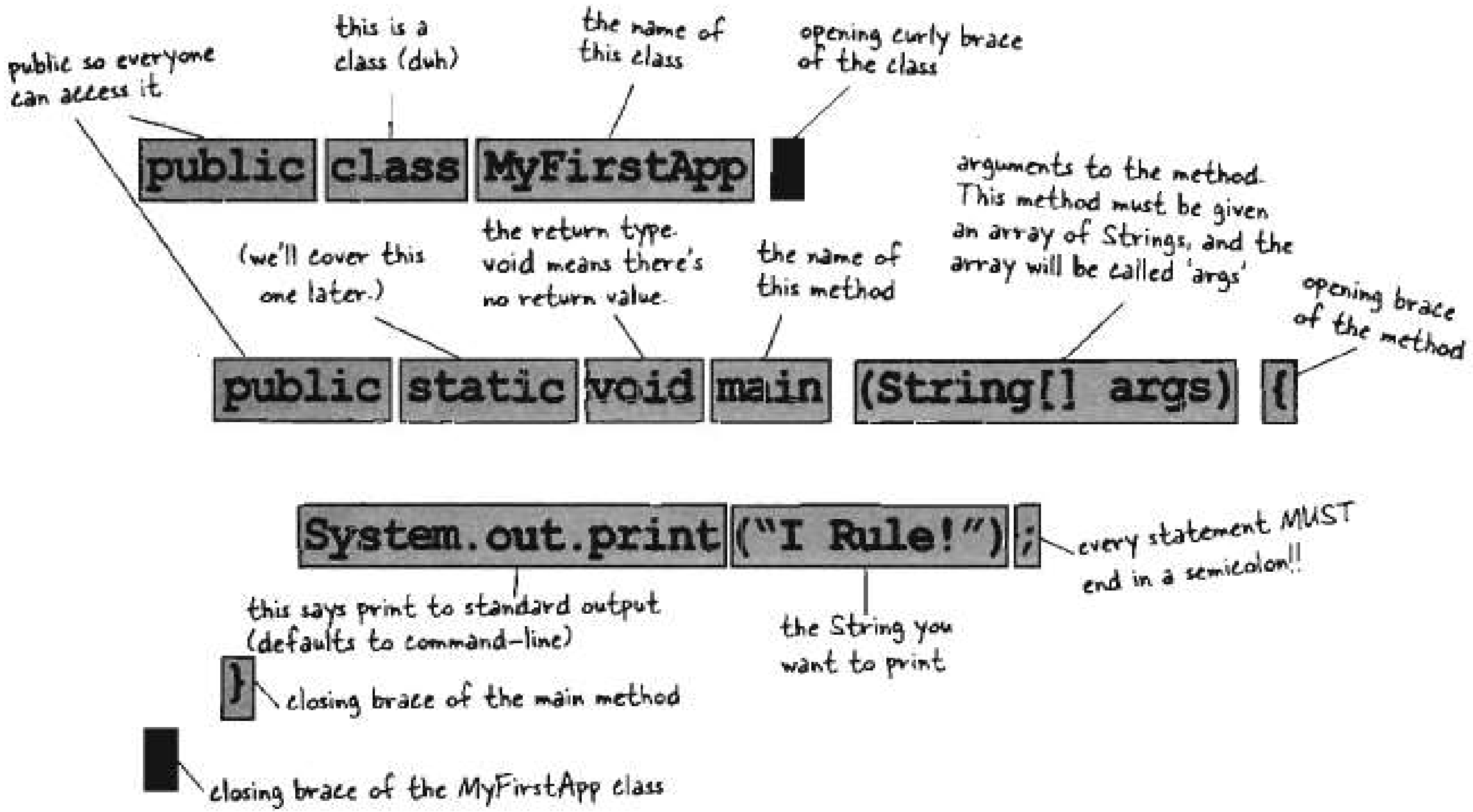
- Save the source code to “HelloWorld.java”
- Compile the source code into Byte code: javac HelloWorld.java
- Run the Byte code: java HelloWorld or java –classpath . HelloWorld

Anatomy of a class

- When the JVM starts running, it looks for the class you give it at the command line. Then it starts looking for a specially-Written method that looks exactly like:

-
  ```
  public static void main (String[] args){
  //  your code goes here
  }
  ```

- Next, the JVM runs everything between the curly braces { }of your main method.
- Every Java application has to have at least one class. and at least one main method (not one main per class ;just one main per *application).*

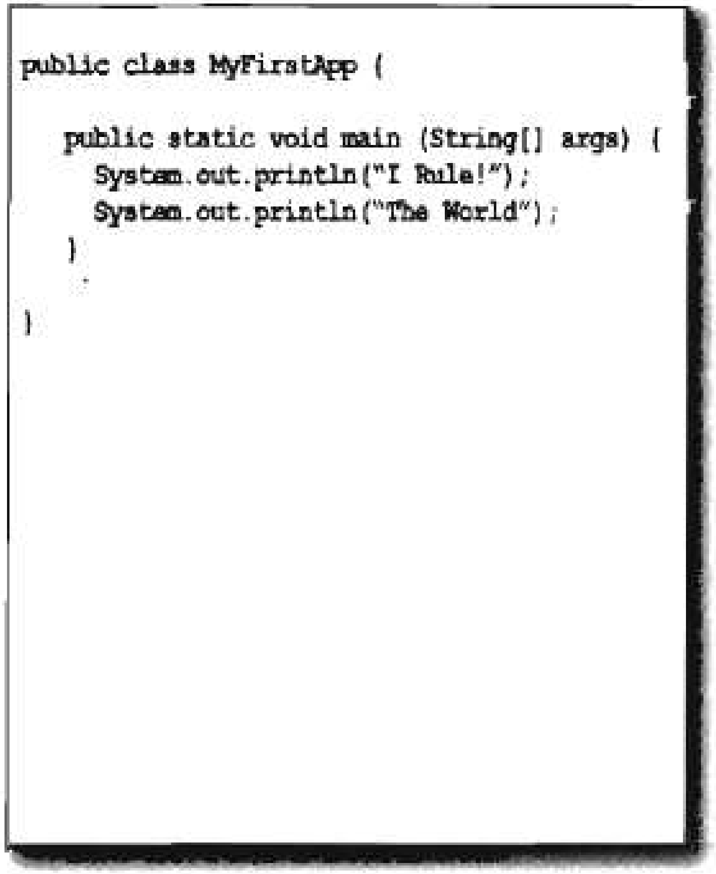


```
public class MyFirstApp
    public static void main (String[] args) {
        System.out.print("I Rule!");
    }
```

public so everyone can access it
this is a class (duh)
the name of this class
opening curly brace of the class
(we'll cover this one later.)
the return type. void means there's no return value.
the name of this method
arguments to the method. This method must be given an array of Strings, and the array will be called 'args'
opening brace of the method
every statement MUST end in a semicolon!!
this says print to standard output (defaults to command-line)
the String you want to print
closing brace of the main method
closing brace of the MyFirstApp class

Writing a class with a main

- In Java, everything goes in a class. You'll type your source code file (with a *.java extension), then compile it into a new class file (with a .class extension).*
- When you run your program, you're really running a *class*.
- Running a program means telling the Java VIrtual Machine (JVM) to "Load the Hello class, then start executing its main () method. Keep running 'til all the code in main is finished.”



```
public class MyFirstApp {

   public static void main (String[] args) {
     System.out.println("I Rule!");
     System.out.println("The World");
   }

}
```

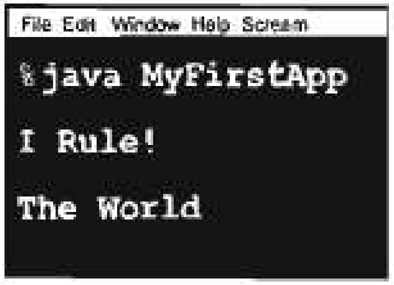
❶ Save

`MyFirstApp.java`

❷ Compile

`javac MyFirstApp.java`

❸ Run



```
% java MyFirstApp
I Rule!
The World
```

Looping and looping and.....

- Java has three standard Looping constructs: *while; do-while, and for.*
- **Simple boolean tests**
- You can do a simple boolean test by checking the value of a variable, using a *comparison operator* including:
- < (less than)
- > (greater than)
- = **=** (equality) (yes, that's *two equals signs)*

Example of a while loop

- public class Loopy {
- public static void main (String[] args) {
- int x = 1;
- System.out.println("Before the Loop");
- while (x < 4) {
- System.out.println("In the loop");
- System.out .prlntln("Value of x is " + x);
- x = x + 1;
- }
- System.out.println("This is after the loop");
- }
- }

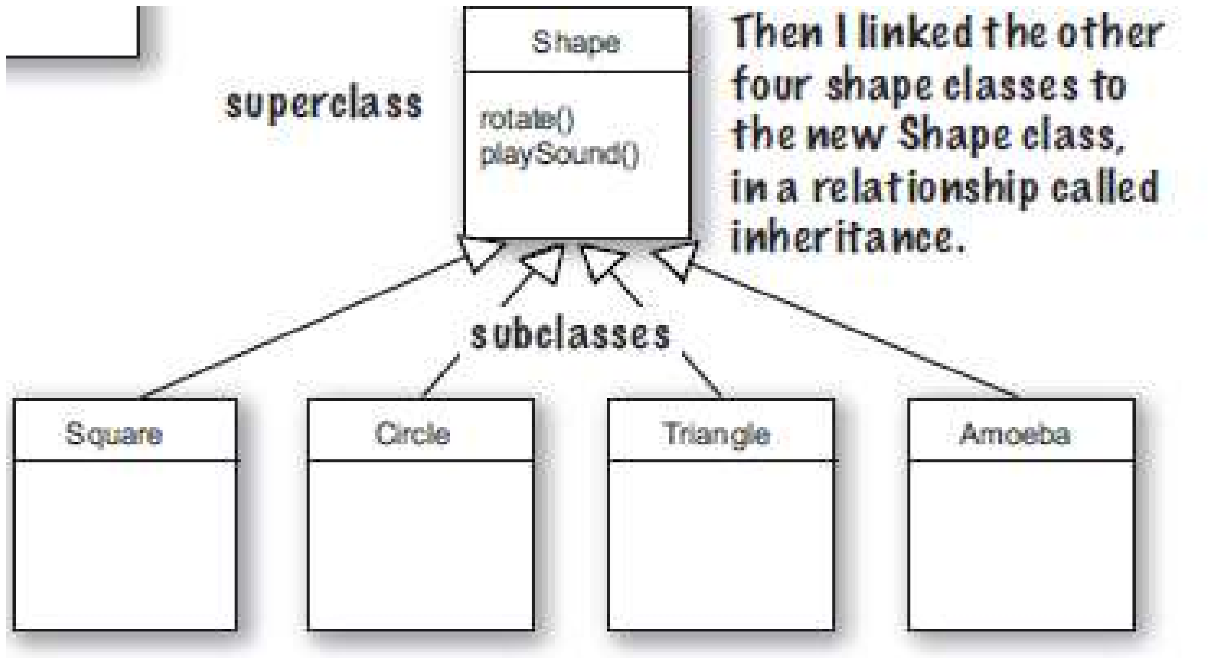
Conditional Branching

- In Java, an *if test is basically the same as the boolean test in a while loop:*
- class IfTest2 {
- public static void main (String[] arqs) {
- int x = 2;
- if (x == 3) {
- System.out.println("x must be *3");*

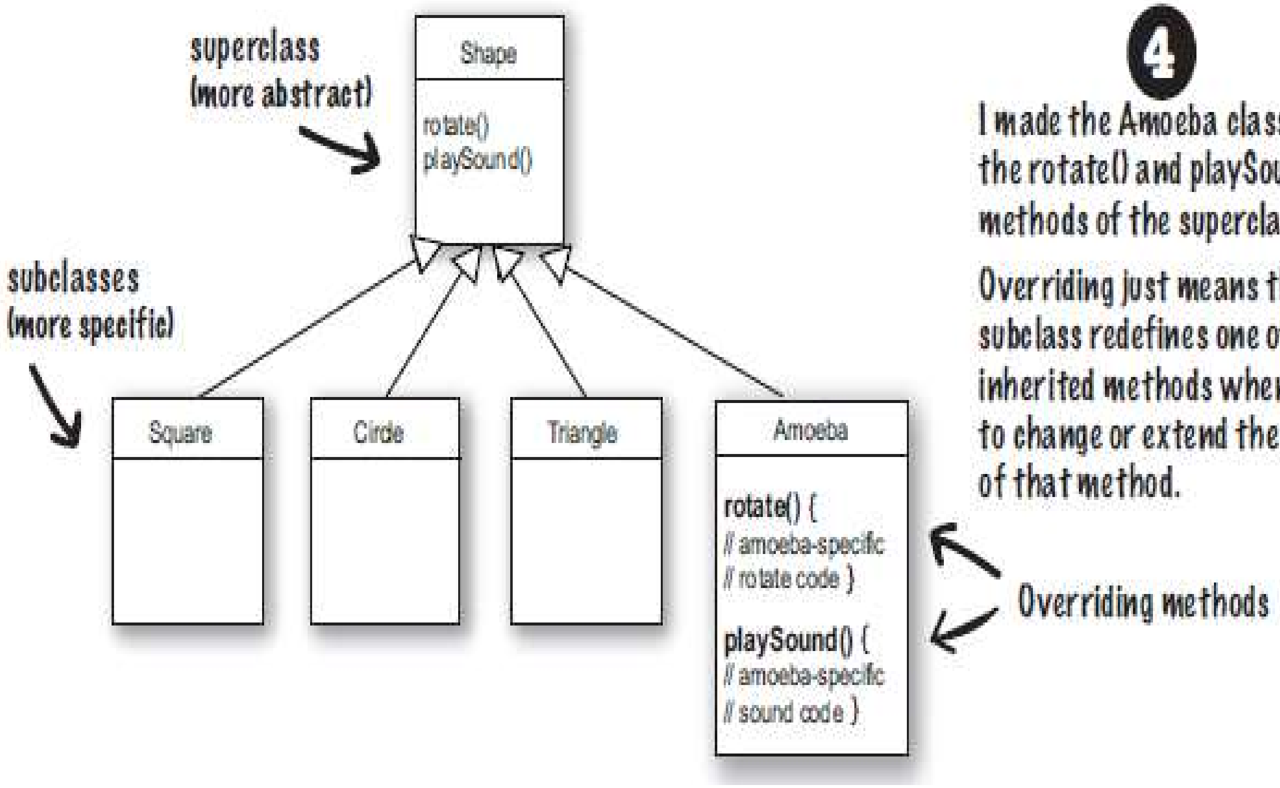
  } else {
- System.out.println("x is NOT 3");
- }
- System.out.println("This runs no matter what ");

  }
- }

3



Then I linked the other four shape classes to the new Shape class, in a relationship called inheritance.



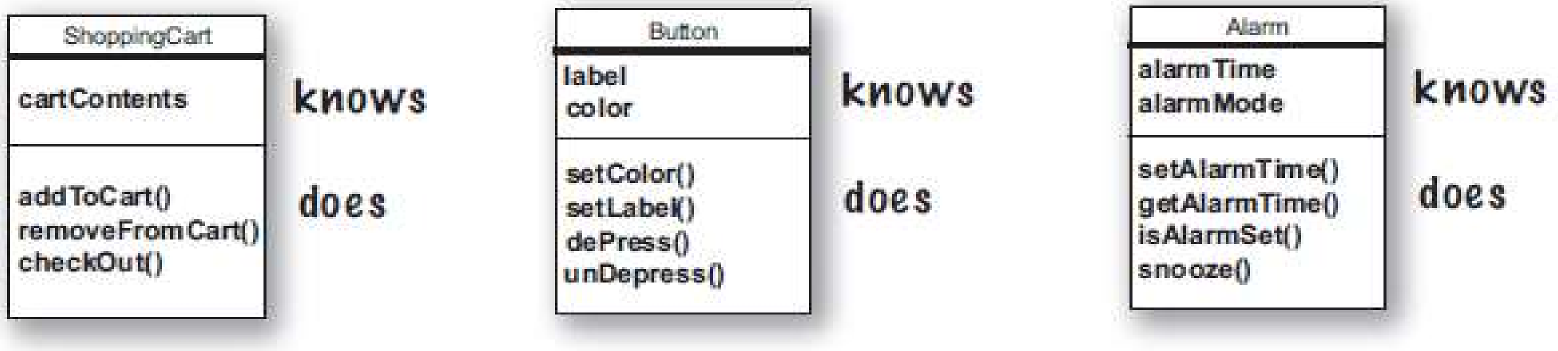
4

I made the Amoeba class override the rotate() and playSound() methods of the superclass Shape.

Overriding just means that a subclass redefines one of its inherited methods when it needs to change or extend the behavior of that method.

When you design a class, think about the objects that will be created from that class type. Think about:

- things the object **knows**
- things the object **does**

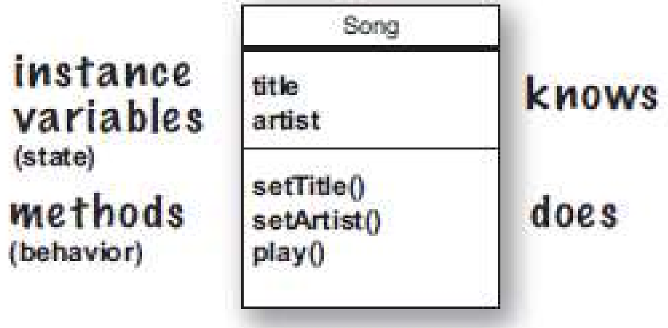


Things an object *knows* about itself are called

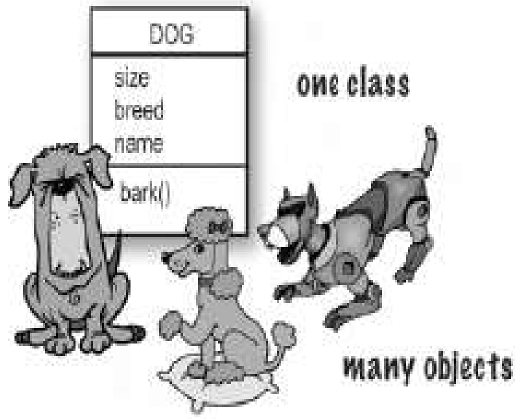
- instance variables

Things an object can *do* are called

- methods

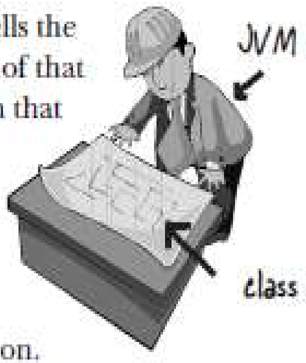


What's the difference between a class and an object?



A class is not an object.

(but it's used to construct them)



A class is a *blueprint* for an object. It tells the virtual machine *how* to make an object of that particular type. Each object made from that class can have its own values for the instance variables of that class. For example, you might use the Button class to make dozens of different buttons, and each button might have its own color, size, shape, label, and so on.

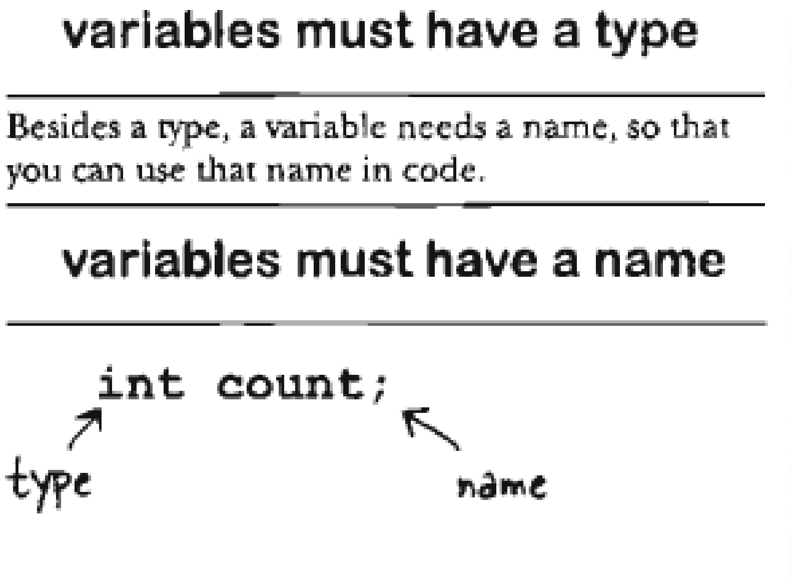
Know Your Variables

- Variables come in two flavors: primitive and reference.
- So *far you've* used variables In two places-as object state (instance variables), and as local variables (variables declared within a *method). Later, we'll use variables as arguments (values sent to a* method by the calling code), and as return types (values sent back to the caller of the method).
- You've seen variables declared as simpie primitive integer vaIues . You've seen variables declared as something more complex like a String or an array.
- But there's gotta be more to life than integers, Strings, and arrays.
- What If you have a PetOwner object with a Dog instance variable? Or a *Car with an Engine? Here we'll unwrap the mysteries of Java* types and look at what you can *declare as a variable,what you can put In a variable, and what you* can *do with a variable.*
- *And we'll finally see what life Is truly like on the garbage-collectible heap.*

Structure overview of Java

- Variables come in two flavors: *primitive and object reference.*
- *Primitives hold* fundamental values (think: simple bit patterns) including integers, Booleans, and floating point numbers.
- Object references hold. well, *references* to *objects.*
- Source code (.java) • Compiled into Byte codes (.class) , as (.exe) in c++
- The Java Application Programming Interface (API)
- A large collection of ready-made software components. It is grouped into libraries of related classes and interfaces; these libraries are known as packages. – Java Virtual Machine (JVM) – Machine code

Primitive Variables:



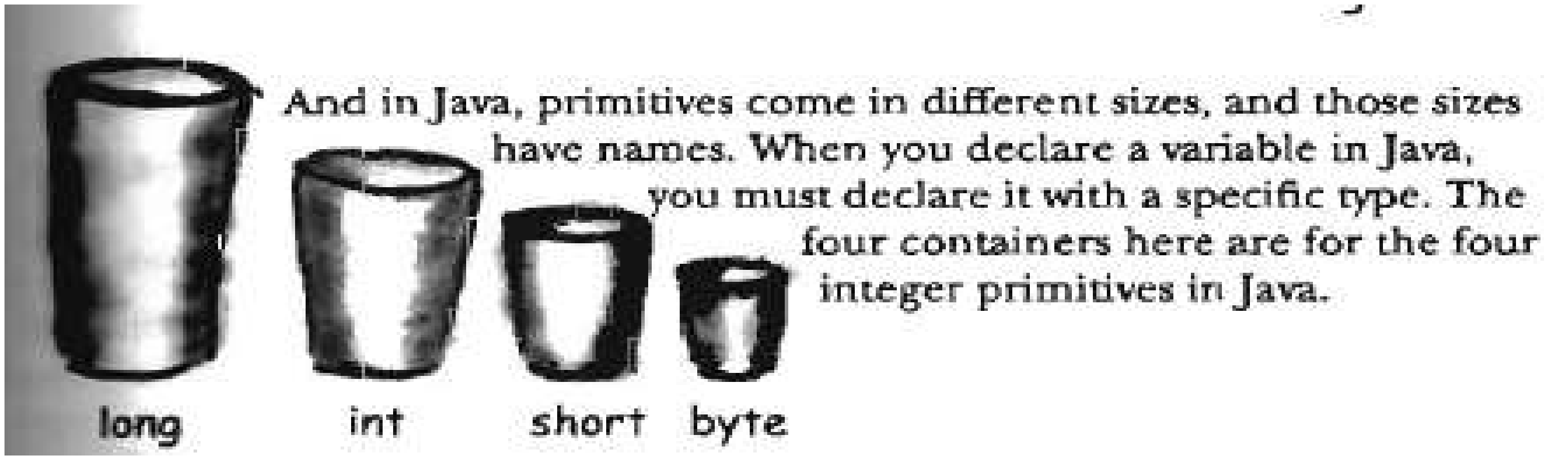
variables must have a type

Besides a type, a variable needs a name, so that you can use that name in code.

variables must have a name

```
int count;
```

- When n you think ofJava variables, think of cups. Coffee cups, tea cups, giant that hold lots and lots of coffee.
- A variable is just a cup. A container. It *holds something*.



And in Java, primitives come in different sizes, and those sizes have names. When you declare a variable in Java, you must declare it with a specific type. The four containers here are for the four integer primitives in Java.

All simple types are always passed by value in Java.

Name	Width	Range
long	64	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
int	32	–2,147,483,648 to 2,147,483,647
short	16	–32,768 to 32,767
byte	8	–128 to 127

Name	Width in Bits	Approximate Range
double	64	4.9e–324 to 1.8e+308
float	32	1.4e–045 to 3.4e+038

Simple Types

- **Character:** Java char is a 16-bit type. The range of a char is 0 to 65,536.
- **Boolean:** Java has a primitive type, called boolean, for logical values. It can have only one of two possible values, true or false.

Variable name Rules:

- It must start with a letter, underscore (_) or dollar sign ($). You can't start a name with a number.
- • After the first character, you can a numbers as well. Just don't start It with a number,
- • It can be anything you like, subject to those two rules, Just so long as It Isn't one of Java's reserved words.

Keywords

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while