

**Chapter
1**

**GETTING
STARTED**

*Get on the
Fast Track!*



**SYS-ED/
Computer
Education
Techniques, Inc.**

Objectives

You will learn:

- Java platform.
- Applets and applications.
- Java programming language: facilities and foundation.
- Memory management and garbage collection.
- Integrated thread synchronization.
- Java components.
- Java object-oriented programming concept.
- Java Virtual Machine.

1 Java Platform

There are several application development platforms and software must be compiled separately to run on each platform. The binary file for an application that runs on one platform cannot run on another platform.

The Java platform is situated on top of the development platforms and executes bytecodes. A program written in the Java Language compiles to a bytecode file that can run wherever the Java Platform is present, on any underlying operating system. Bytecodes are not specific to any physical machine; they are machine instructions for a virtual machine. The same exact file can run on any operating system that is running the Java Platform.

- Developers can write object-oriented, multithreaded, dynamically linked applications using the Java language. The platform has built-in security, exception handling, and automatic garbage collection.
- JIT: Just-in-time compilers are available to speed up execution by converting Java bytecodes into machine language.
- From within the Java Language, developers can also write and call native methods in C, C++ or another language, compiled to a specific underlying operating system-for speed or special functionality.

2 Applets and Applications

The Java Platform has two fundamental parts:

• Java Virtual Machine	• Java Application Programming Interface (Java API Applets and Applications)
------------------------	--

The Java Platform enables developers to create two different kinds of programs:

• Applets	• Applications
-----------	----------------

2.1 Applets

Applets are programs that require a browser to run.

- The <applet> tag is embedded in a web page and names the program to be run.

When that page is accessed by a user, either over the Internet or corporate intranet, the applet automatically downloads from the server and runs on the client machine.

Because applets are downloaded, they tend to be designed small or modular, to avoid large download times.

2.2 Applications

Applications are programs that do not require a browser to run; they have no built-in downloading mechanism. When an application is called, it runs. In this way, applications are just like programs in other languages.

- An applet requires a network to run, while an application does not.
- Applications have greater freedom in that they have full access to system services.

3 Java Programming Language: Facilities and Foundation

The Java programming language:

- is object-oriented (with single inheritance).
- is statically typed.
- is multithreaded.
- is dynamically linked.
- has automatic garbage collection.

The Java programming language syntax is based on C and C++. However, there is less redundancy with Java than with these other programming languages.

For numeric programming, the Java Language has platform-independent data types:

- array bounds-checking.
- well-defined IEEE arithmetic.

These capabilities provide a foundation for writing stable numerical algorithms that give repeatable results. Expressions, statements, and operators in the Java Language are in most cases the same as in the C language.

The Java language facilitates the discovery of bugs early, during development, before the software is released.

This is achieved by:

- strong data typing.
- automatic garbage collection.
- array bounds checking.
- lack of the pointer data type.

The Java language has multithreading built in; it is a foundation model based on the synchronization of thread-critical code which will avoid racing or timing problems.

4 Object Oriented

Java is an object oriented programming language and has the following advantages:

- The development cycle is much faster because Java technology is interpreted.
- The compile-link-load-test-crash-debug cycle is obsolete; programs need only to be compiled and run.
- Applications are portable across multiple platforms. Once an application has been coded; it will never have to be ported. The application will run without modification on multiple operating systems and hardware architectures.
- Java applications are efficient because the Java runtime environment manages the memory.
- Interactive graphical applications will perform efficiently because multiple concurrent threads of activity in a application are supported by the multithreading built into the Java programming language and runtime platform.
- Java applications are adaptable to changing environments because code modules can be dynamically downloaded from anywhere on the network.
- Applications downloaded from the Internet will be secure; the Java runtime environment has built-in protection against viruses and tampering.

5 Data Type

Other than the primitive data types, everything in the Java programming language is an object.

The Java programming language follows C and C++ fairly closely in its set of basic data types, with a couple of minor exceptions.

There are three groups of primitive data types:

• Numeric types	• Character types	• Boolean types
-----------------	-------------------	-----------------

5.1 Numeric Data Types

Integer numeric types are 8-bit byte, 16-bit short, 32-bit int, and 64-bit long.

The 8-bit byte data type in Java has replaced the old C and C++ char data type.

Java places a different interpretation on the char data type.

- There is no unsigned type specifier for integer data types in Java. Real numeric types are 32-bit float and 64-bit double.
- A floating point literal value, like 23.79, is considered double by default; it must explicitly be cast to float if you wish to assign it to a float variable.

5.2 Character Data Types

Java language character data is a departure from traditional C. Java's char data type defines a sixteen-bit Unicode character. Unicode characters are unsigned 16-bit values that define character codes in the range 0 through 65,535.

If a declaration is written such as:

```
char myChar = 'Q';
```

a Unicode (16-bit unsigned value) type is initialized to the Unicode value of the character Q.

5.3 Boolean Data Types

Java added a Boolean data type as a primitive type.

A Java Boolean variable assumes the value true or false. A Java programming language Boolean is a distinct data type; unlike common C practice, a Java programming language Boolean type can't be converted to any numeric type.



6 Arithmetic and Relational Operators

All the familiar C and C++ operators apply.

The Java programming language has no unsigned data types, so the >>> operator has been added to the language to indicate an unsigned (logical) right shift.

Java also uses the + operator for string concatenation.



7 Arrays

In contrast to C and C++, the Java programming language arrays are first-class language objects.

An array in the Java programming language is a real object with a run-time representation. Arrays can be declared and allocated of any type; arrays of arrays can be used for obtaining a set of arrays to obtain multi-dimensional arrays.

An array can be declared with a declaration such as:

```
int myStuff[];
```

This code states that myPoints is an uninitialized array of Points. At this time, the only storage allocated for myPoints is a reference handle. At some point, the amount of required storage will need to be allocated such as:

```
myStuff = new int[10];
```

This will allocate an array of ten references.

7.1 String

Strings are Java programming language objects, not pseudo-arrays of characters such as in C.

The Java compiler understands that a string of characters enclosed in double quote signs is to be instantiated as a String object.

The declaration:

```
String hello = "Hello world!";
```

instantiates an object of the String class behind the scenes and initializes it with a character string containing the Unicode character representation of "Hello world!".

Java technology has extended the meaning of the + operator to indicate string concatenation.

Example:

```
System.out.println("There are " + num + " characters in the file.");
```

This code fragment concatenates the string "There are " with the result of converting the numeric value num to a string, and concatenates that with the string " characters in the file."

Then it prints the result of those concatenations on the standard output.

8 Memory Management and Garbage Collection

C and C++ programmers explicitly manage memory by:

- allocating memory.
- freeing memory.
- keeping track of what memory can be freed when.

Explicit memory management typically is directly related to:

• bugs	• crashes	• memory leaks	• poor performance.
--------	-----------	----------------	---------------------

Java technology removes the burden of memory management load from the programmer. C-style pointers, pointer arithmetic, malloc, and free do not exist.

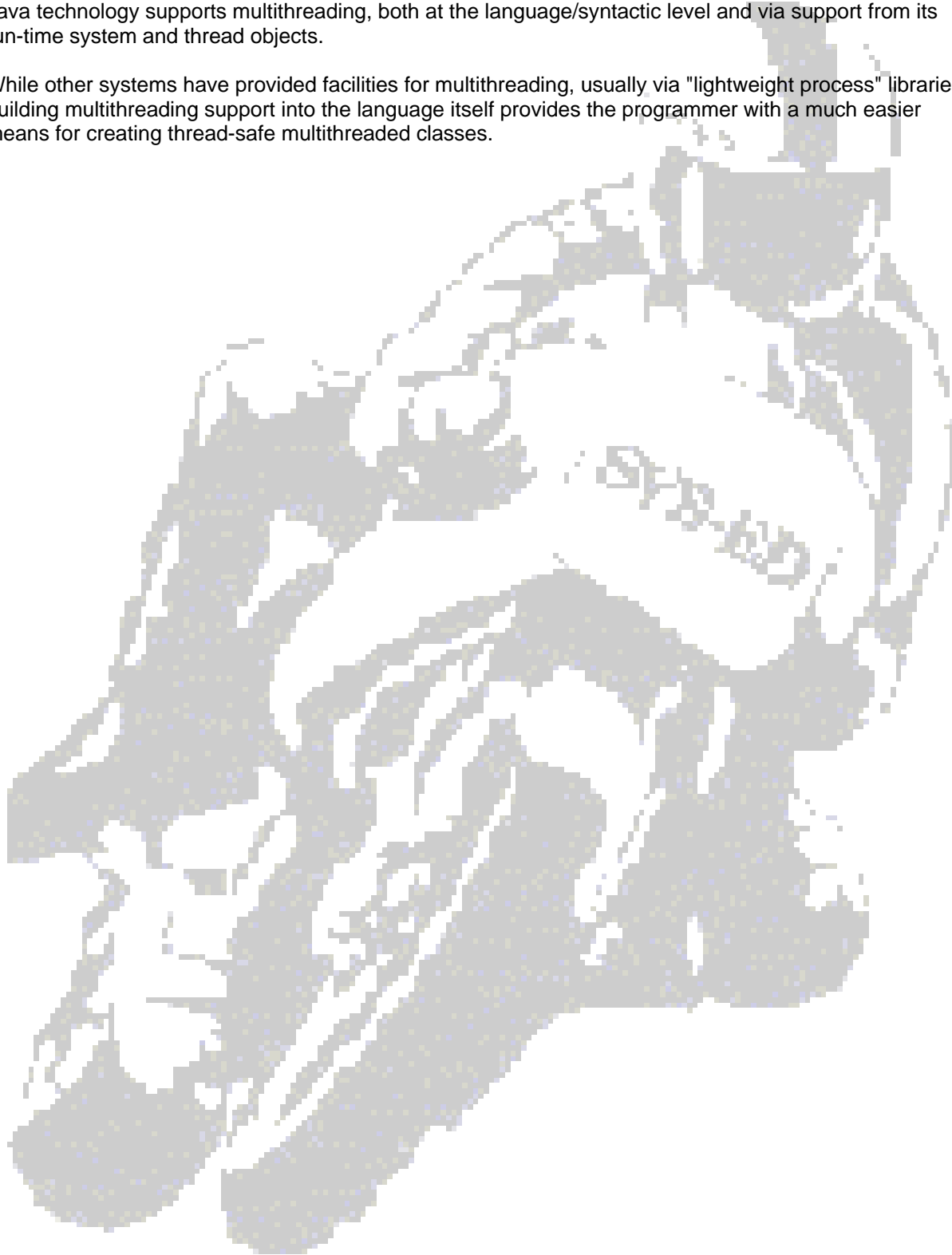
Automatic garbage collection is an integral part of Java and its run-time system. Once an object has been allocated the run-time system keeps track of the object's status and automatically reclaims memory when objects are no longer in use, freeing memory for future use.

Java technology's memory management model is based on objects and references to objects. Java technology has no pointers.

9 Integrated Thread Synchronization

Java technology supports multithreading, both at the language/syntactic level and via support from its run-time system and thread objects.

While other systems have provided facilities for multithreading, usually via "lightweight process" libraries, building multithreading support into the language itself provides the programmer with a much easier means for creating thread-safe multithreaded classes.



10 Java Featureset

Java:

- Is both compiled and interpreted; this serves to help make it both fast and platform-independent.
- Is an object-oriented language which utilizes class structure and how and why classes are extended.
- Utilizes the Java Virtual Machine for executing a program and providing security.
- Is augmented by the Java API. Created by Sun Microsystems the Java API has a number of classes, which serve to make writing Java programs faster and easier.

10.1 Java: Interpreted Language

Java is an interpreted language which is compiled. Only about 20 percent of the Java code is interpreted by the browser. Both Java's security and its ability to run on multiple platforms are derived from the final steps of compilation which are handled locally.

A programmer first compiles Java source into bytecode using the Java compiler. These bytecodes are binary and architecturally neutral. However, the bytecode is not complete until it is put together with a Java runtime environment which is usually a browser.

Since each Java runtime environment is for a specific platform, the bytecodes can be interpreted for the specific platform and the final product will work on that specific platform.

10.2 Java: Object-Oriented Language

Java is an object-oriented language. It is part of a family of languages that focus on defining data as objects and the methods that may be applied to those objects. An object-oriented programming language describes interactions among data objects.

Many OOP languages support multiple inheritance, which can sometimes lead to confusion or unnecessary complications. Java doesn't. Java supports only single inheritance. That means each class can inherit from only one other class at any given time. This type of inheritance avoids the problem of a class inheriting classes whose behaviors are contradictory or mutually exclusive.

Java provides the capability for creating totally abstract classes known as interfaces. Interfaces allow methods to be defined which can be shared with several classes, without regard for how the other classes are handling the methods.

10.3 Java Virtual Machine

The JVM: Java Virtual Machine is a virtual computer that resides in memory only. The JVM allows Java programs to be executed on a variety of platforms as opposed to only the one platform for which the code was compiled. The fact that Java programs are compiled for the JVM is what makes the language so unique.

In order for Java programs to run on a particular platform, the JVM must be implemented for that platform.

All processors use registers; the JVM uses the following registers to manage the system stack:

Program counter	Keeps track of where exactly the program is in execution.
Optop	Points to the top of the operand stack.
Frame	Points to the current execution environment.
Vars	Points to the first local variable of the current execution environment.

The Java development team decided that Java would only use four registers because if Java had more registers than the processor it was being ported to, then that processor would take a serious reduction in performance.

The stack is where parameters are stored in the JVM. The JVM is passed the bytecode from the Java program and creates a stack frame for each method.

Each frame holds three kinds of information:

Local variables	An array of 32-bit variables that is pointed to by the vars register.
Execution environment	Where the method is executed and is pointed to by the frame register.
Operand stack	Acts on the first-in, first-out principle, or FIFO. It is 32 bits wide and holds the arguments necessary for the opcodes. The top of this stack is indexed by the optop register.

11 Garbage-Collection Heap

The heap is the collection of memory from which class instances are allocated. Any time memory is allocated with the new operator, that memory comes from the heap.

The garbage collector can be called directly, but it is not necessary or recommended under most circumstances. The runtime environment keeps track of the references to each object on the heap and automatically frees the memory occupied by objects that are no longer referenced.

Garbage collections run as a thread in the background and clean up during CPU inactivity.

12 OOP: Object Oriented Programming

In OOP programming, the methods for the car to run and the actual running of the car are combined into one object:

```
public class Car{
    int weight;
    float speed;
    int hp;
    double dragCoef;

    public void speedUp(){
        speed += hp/weight;
    }

    public void slowDown(){
        speed -= speed * dragCoef;
    }

    public void stop(){
        speed=0;
    }
}
```

Within each of the new methods, there is no need to either reference the variables using dot notation or pass in a reference to variables. The methods implicitly know about the variables of their own class. These variables are also known as field variables.

12.1 Extending Objects Through Inheritance

The next step in developing objects is to create multiple objects based on one super object. Inheritance is a feature of OOP programming that provides this capability for achieving this.

Race Cars

```
class Lamborghini extends Car{
    public void superCharge(){
        for (int x=0;x<infinity;x++)
            speedUp();
    }
}

class Volvo extends Car{
    CDPlayer cd;

    public void goFaster(){
        while(I_Have_Gas){
            speedUp();
        }
    }

    public void jam(){
        cd.turnOn();
    }
}
```

Race Track

```
class RaceTrack {
    Car  theCars[] = new Car[3];
    int  numberOfCars = 0;
    public void addCar(Car newCar){
        theCars[numberOfCars]=newCar;
        numberOfCars++;
    }

    public void yellowLight(){
        for (int x=0;x<numberOfCars;x++)
            theCars[x].slowDown();
    }
}
```

12.2 Program

```
Class RaceProgram{
    Lamborghini me = new Lamborghini();
    Volvo      you = new Volvo();
    RaceTrack rc = new RaceTrack();
    public void start(){
        rc.addCar(me);
        rc.addCar(you);
        while(true){
            if (somethingIsWrong)
                rc.yellowLight();
        }
    }
}
```

13 Objects as Multiple Entities

A pitfall for the procedural programmer is conceptualizing of the data in a program as a fixed quantity.

For the OOP programmer, it is necessary to deal with objects. And regardless as to whether there is a single or one hundred objects, it doesn't affect the program in any way.

