Java’s program control statements can be put into the following categories: selection, iteration, and jump. *Selection* statements allow your program to choose different paths of execution based upon the outcome of an expression or the state of a variable. *Iteration* statements enable program execution to repeat one or more statements (that is, iteration statements form loops). *Jump* statements allow your program to execute in a nonlinear fashion. All of Java’s control statements are examined here.

Java supports two selection statements: if and switch.

if (*condition*) *statement1*;

else *statement2*;

Nested ifs

A *nested* if is an if statement that is the target of another if or else.

if(i == 10) {

if(j < 20) a = b;

if(k > 100) c = d; // this if is

else a = c; // associated with this else

}

else a = d;

The if-else-if Ladder

if(*condition*)

*statement*;

else if(*condition*)

*statement*;

else if(*condition*)

*statement*;

.

.

.

else

*statement*;

The if statements are executed from the top down.

switch

The switch statement is Java’s multiway branch statement. It provides an easy way to

dispatch execution to different parts of your code based on the value of an expression.

switch (*expression*) {

case *value1*:

// statement sequence

break;

case *value2*:

// statement sequence

break;

.

.

.

case *valueN* :

// statement sequence

break;

default:

// default statement sequence

}

For versions of Java prior to JDK 7, *expression* must be of type byte, short, int, char, or an

enumeration. (Enumerations are described in Chapter 12.) Beginning with JDK 7, *expression* can also be of type String. Each value specified in the case statements must be a unique

constant expression (such as a literal value). Duplicate case values are not allowed. The type

of each value must be compatible with the type of *expression*.

The break statement is optional. If you omit the break, execution will continue on into the

next case. It is sometimes desirable to have multiple cases without break statements between

them.

Nested switch Statements

switch(count) {

case 1:

switch(target) { // nested switch

case 0:

System.out.println("target is zero");

break;

case 1: // no conflicts with outer switch

System.out.println("target is one");

break;

}

break;

case 2: // ...