Algorithms

To solve a problem with a computer program, you must:

- plan each step in careful detail;
- understand Java's building blocks.

A procedure for solving a problem in terms of:

- the actions to be executed
- the order in which they are executed

is called an algorithm.

For example, a simple "eat breakfast" algorithm might be:

1. boil an egg;
2. eat an egg.

Clearly, executing these actions in reverse order would be less happy.

Pseudocode

- Pseudocode is an informal language programmers use to describe an algorithm.
- It is a halfway point between English and Java.
- Why use it? To get the overall structure of your algorithm right without having to worry about certain programming details.
- Some kinds of pseudocode have a graphical (flowchart) flavor.
- Ours is entirely textual.

Control Structures

- Often, we want a sequence of Java statements to be executed in the order in which they are written: sequentially.
- Sometimes, we want the order of execution to depend on the value of some piece of data.

- In this case, we use a control structure.
- Such structures control the flow of execution of the statements.
- Such structures begin with a key word (e.g., if, while, for).
- There are 2 varieties:
  - Selection structures (if, if/else, and switch)
  - Repetition structures (while, do/while, and for)
- Thus, Java has only 7 control structures (including sequential).

The if Selection Structure

- A selection structure selects a statement to execute from a set (of 1 or more executable statements).

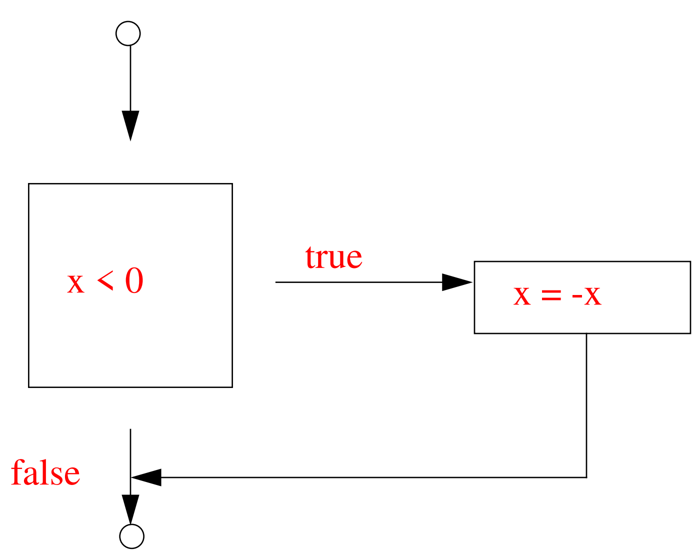
  ...

  if x is less than 0,

      replace x with -x;

- The statement that is executed if the condition is true is indented, indicating to the reader (not the compiler) that its execution is conditional.
- The Java translation of the pseudocode above is:

```
if (x < 0)
    x = -x;
```



- The flowchart representation of the if statement has a single entry point (of flow) and a single exit.
- This is true of all Java control structures, making it easy to stack them like blocks.
- The condition must be a boolean expression: one whose value is either true or false.
- A compiler error results, if this is not the case.
- All control structures in Java are represented with just 2 shapes:
  - rectangles (representing actions)
  - diamonds (representing decisions)

  graphically illustrating a decision/action model of programming.

The if/else Selection Structure

The pseudocode if/else statement:

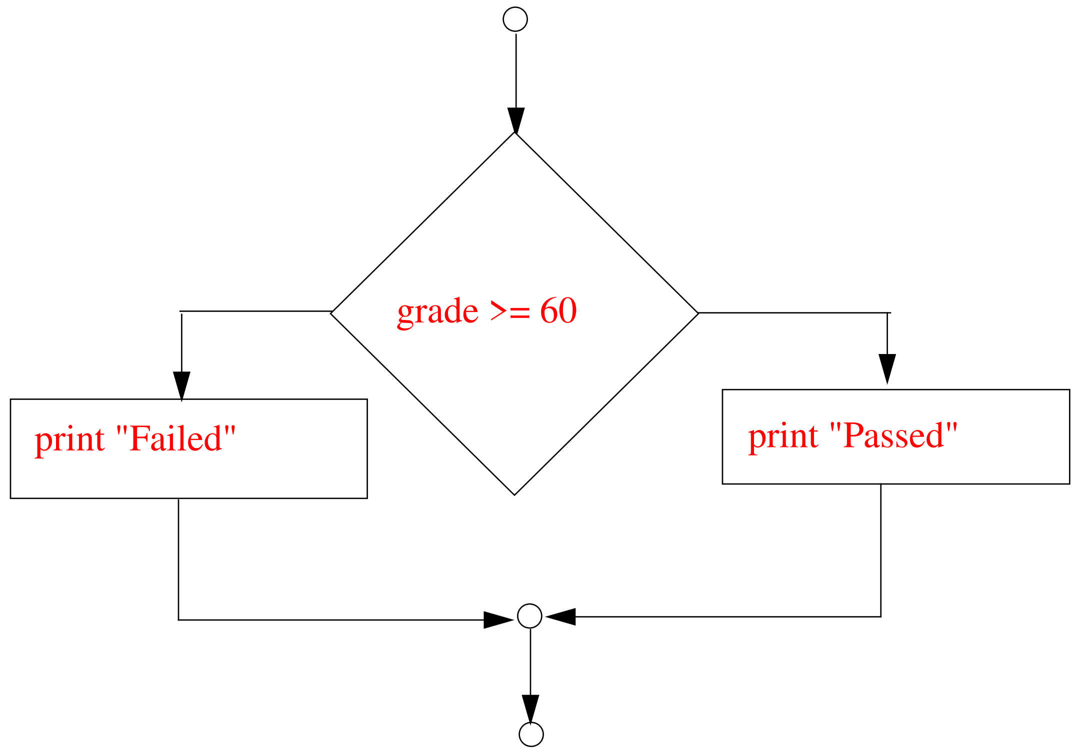
```
if student's grade is greater than 60
    print "Passed"
else
    print "Failed"
```

provides for selecting among 2 alternatives, depending on whether the condition evaluates to true or false.

A Java translation is:

```
if (grade > 60)
    System.out.println("Passed");
else
    System.out.println("Failed");
```

This is represented as a flowchart below.



The conditional operator

- The conditional operator (?:) is similar to the if/else control structure.
- It is Java's only ternary operator (i.e., it has 3 operands).
- The operator with its 3 operands form a conditional expression.
- The 1st operand is a boolean expression;
- The 2nd operand is the expression's value if the condtion is true;
- The 3rd operand is the expression's value if the condtion is false;

```
System.out.println( (grade >= 60) ? "Passed" : "Failed");
```

What does the above statement do?

Nested if/else structures

```
if 2 divides x
    print "x is even"
else
    if 3 divides x
        print "x is odd; 3 divides x"
    else
        if 5 divides x
            print "2,3 don't; 5 does"
        else
            print "x is relatively prime to 30"
```

A Java translation is:

```
if (x % 2 == 0)
    System.out.println("x is even");
else
    if (x % 3 == 0)
        System.out.println("x is odd; 3 divides x");
    else
        if (x % 5 == 0)
            System.out.println("2,3 don't; 5 does");
        else
            System.out.println("x relatively prime to 30");
// quoted string should NOT be broken over a line
```

- The successive indenting indicates the successive nestings of if/else statements;
- It makes it hard to avoid breaking executable statements over a line.
- Most Java programmers lay out this 4-way selection as follows:

```
if (x % 2 == 0)
    System.out.println("x is even");
```

```
else if (x % 3 == 0)
    System.out.println("x is odd; 3 divides x");
else if (x % 5 == 0)
    System.out.println("2,3 don't; 5 does");
else
    System.out.println("x relatively prime to 30");
```

- The Java compiler associates an else with the previous if unless the programmer indicates otherwise with braces:

```
if (x > 5)
    if (y > 5)
        System.out.println("x and y are > 5");
else
        System.out.println("x is <= 5");
```

- "x is <= 5" gets printed only when x > 5 and y <= 5.
- To get what the programmer apparently intended here:

```
if (x > 5)
    {
    if (y > 5)
        System.out.println("x and y are > 5");
    }
else
        System.out.println("x is <= 5");
```

- To include more than 1 statement in the body of an if, use braces to create a compound statement.

```
if (grade >= 60)
    System.out.println("Passed");
else
    {
```

```
    System.out.println("Failed");
    System.out.println("Please repeat the course");
    }
```

- Without braces, the string "Please repeat the course" would be printed regardless of grade's value.
- A compound statement may occur any where a single statement can occur.
- To avoid a missing closing brace, when you type the opening brace, immediately type the closing brace, then put statements within them.
- An empty statement (;) can occur almost any place a single statement can occur.
- The following cause compilers, however:

```
if (x > 5)
    ; // compiler error
if (x > 5)
    x = x + 1;
else
    ; // compiler error
```

- The compiler is saying: “Tell me what to do, or shut up.”
- This may seem impatient, but it’s really a favor.

The while Repetition Structure

- The pseudocode statement

```
While there are more items on my grocery list
    Purchase the next item
    Cross the item off the list
```

- In this repetition structure the condition is tested

  there are more items on my grocery list

  While it is true, the body of the while statement is executed

```
    Purchase the next item
    Cross the item off the list
```

- Clearly, the body will be executed indefinitely unless something in the body of the while changes the value of the condition.
- In the case above, it is the statement

  Cross the item off the list
- Always check to see that you have such a statement in the body of your while statements.

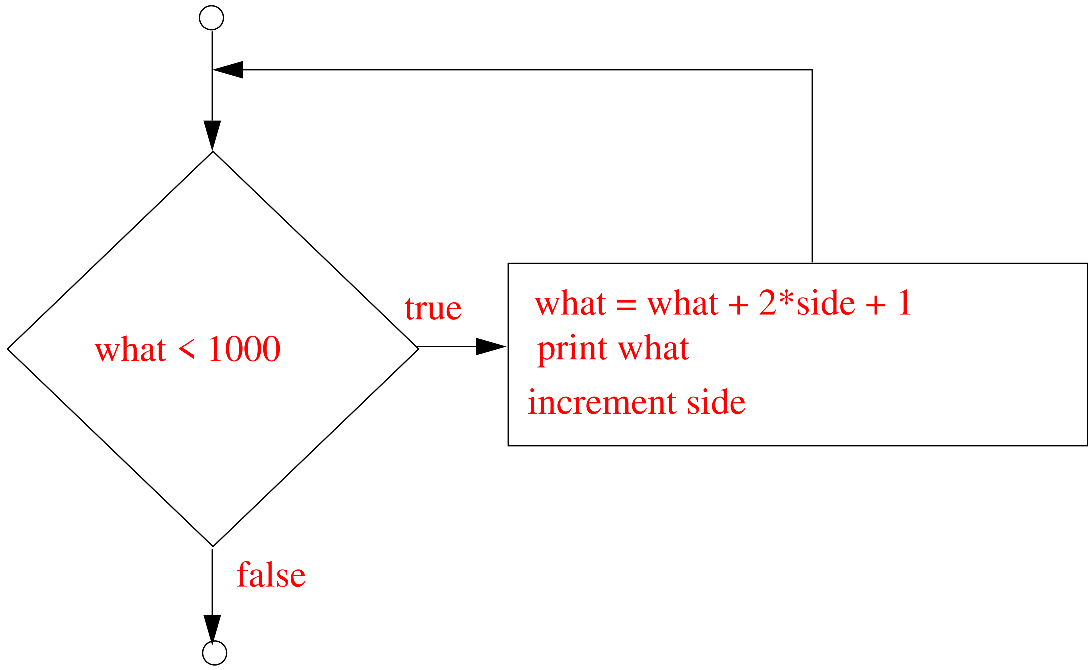
```
int side = 0;
int what = 0;
while (what < 1000) //compute,print squares
    {
    what = what + 2*side + 1;
    System.out.println("what = " + what);
    side = side + 1;
    }
```

- When executed, what will be printed?
- The flow chart representation of a while looks like:
- The condition of the while is tested before the body is executed.
- If the condition initially is false, then the body is never executed.

Formulating Algorithms: Case Study 1 (Counter-Controlled Repetition)

- Here's the problem:

  A class of 10 students took a quiz. The grades (1 of the letters A, B, C, D, or F) for this quiz are available to you. Determine the class average on the quiz.
- A pseudocode algorithm for this follows:




```
//Give variables an initial value
Set total to 0
Set grade counter to 1

While grade counter <= 10
    {
    Input the next grade

    If the letter grade equals A
        Add 4 to total
    else if the letter grade equals B
        Add 3 to total
    else if the letter grade equals C
        Add 2 to total
    else if the letter grade equals D
        Add 1 to total
    else if the letter grade equals F
        Add 0 to total

    Increment grade counter by 1
    }
Set the class average to total/10
```

Print the class average

- The compiler gives you an error message if you use a variable in an expression (refer to its value) before giving it one.

```
import java.io.*;
```

- This line imports the Java I/O package, wherein System.in and System.out methods are defined.

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

- When creating an application, we still define a class (but it does not extend Applet).
- The class created must have a main method.
- For now, just copy this line for every application you create.