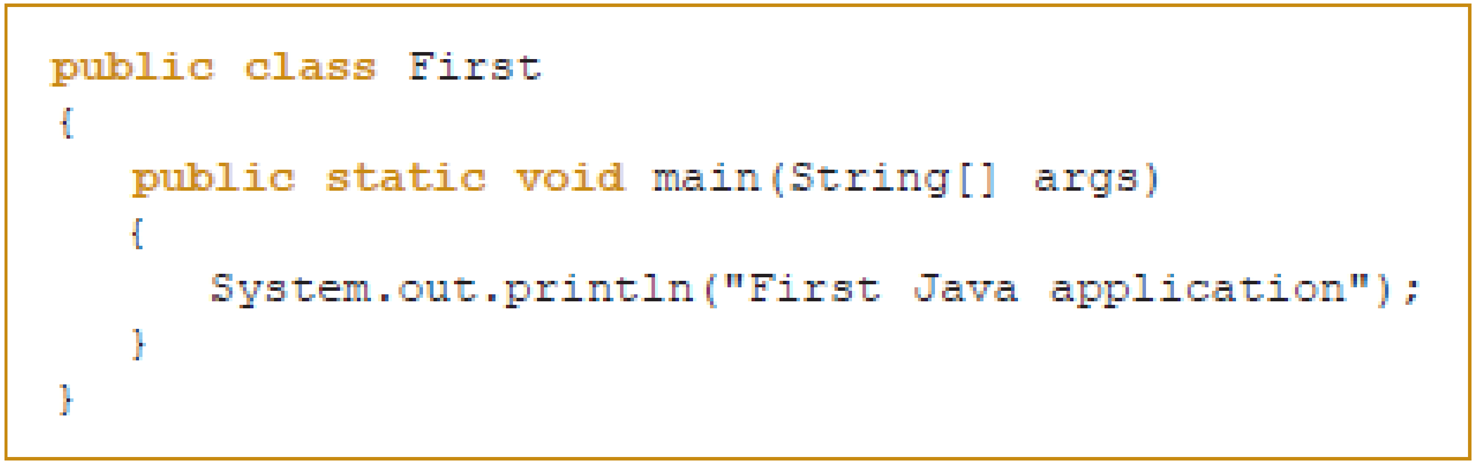
Unit 2
Intro to Java

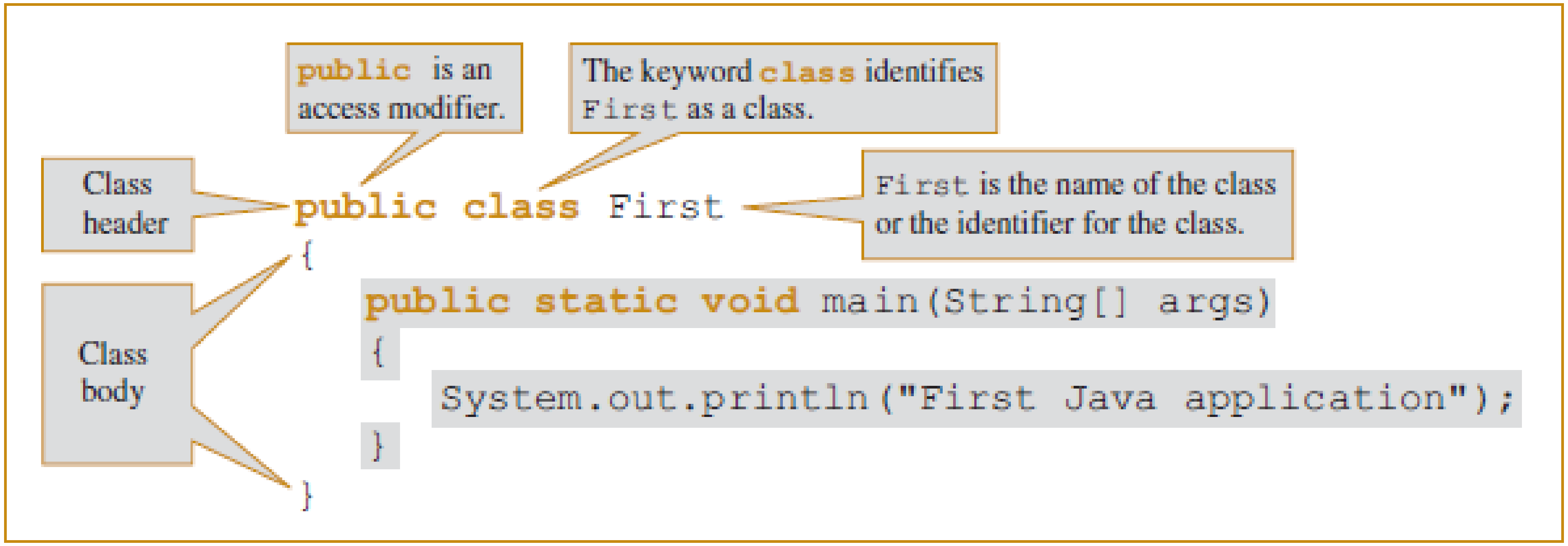
Dr. Magdi AMER

First Program



```
public class First
{
    public static void main(String[] args)
    {
        System.out.println("First Java application");
    }
}
```

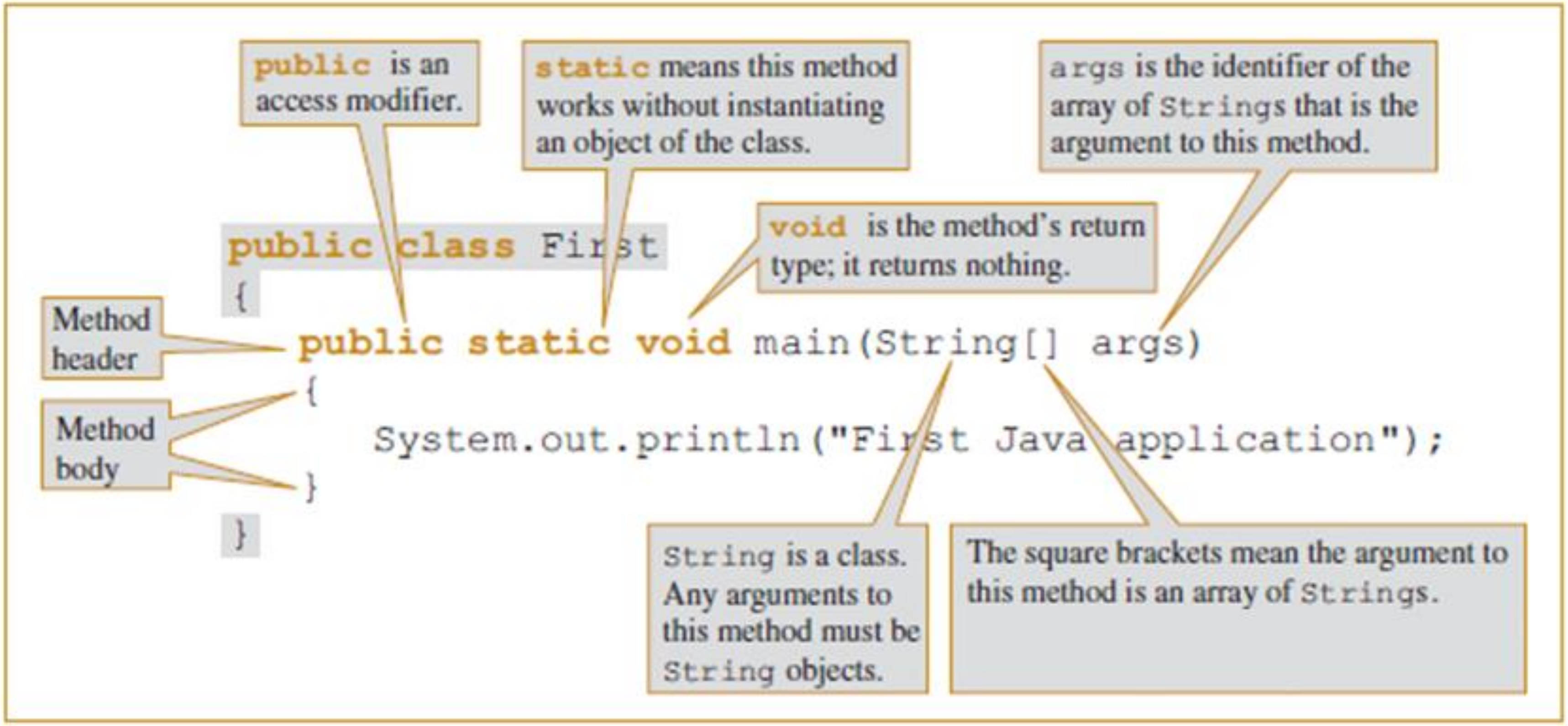
First Program



```
public class First
{
    public static void main(String[] args)
    {
        System.out.println("First Java application");
    }
}
```

public is an access modifier.
The keyword class identifies First as a class.
Class header
First is the name of the class or the identifier for the class.
Class body

First Program



```
public class First
{
    public static void main(String[] args)
    {
        System.out.println("First Java application");
    }
}
```

public is an access modifier.

static means this method works without instantiating an object of the class.

args is the identifier of the array of Strings that is the argument to this method.

void is the method's return type; it returns nothing.

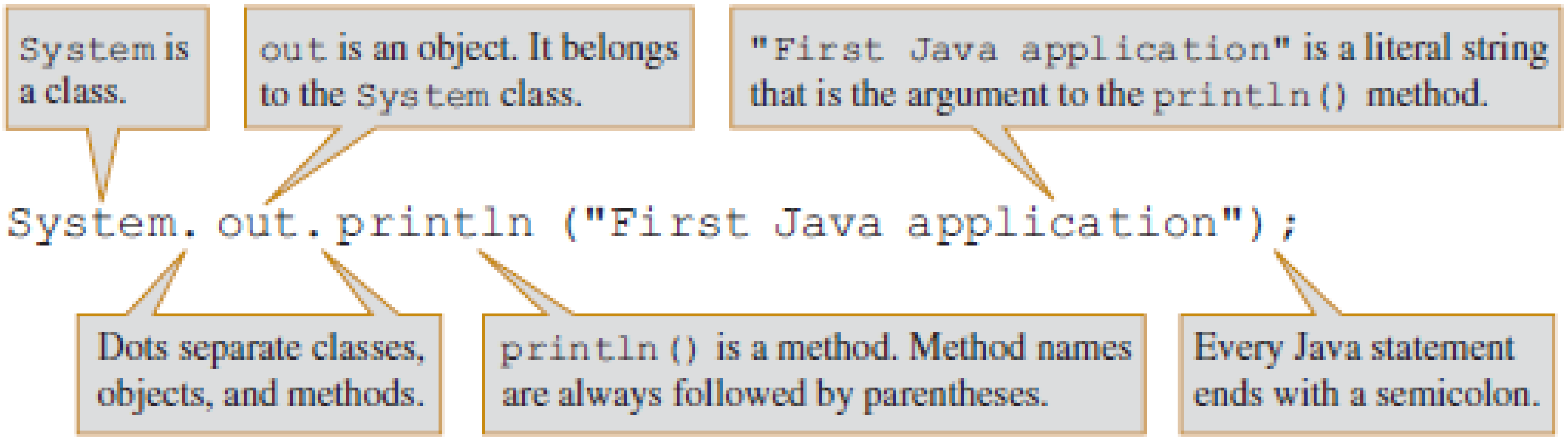
Method header

Method body

String is a class. Any arguments to this method must be String objects.

The square brackets mean the argument to this method is an array of Strings.

First Program


System is a class.
out is an object. It belongs to the System class.
"First Java application" is a literal string that is the argument to the println() method.
System. out. println ("First Java application");
Dots separate classes, objects, and methods.
println() is a method. Method names are always followed by parentheses.
Every Java statement ends with a semicolon.

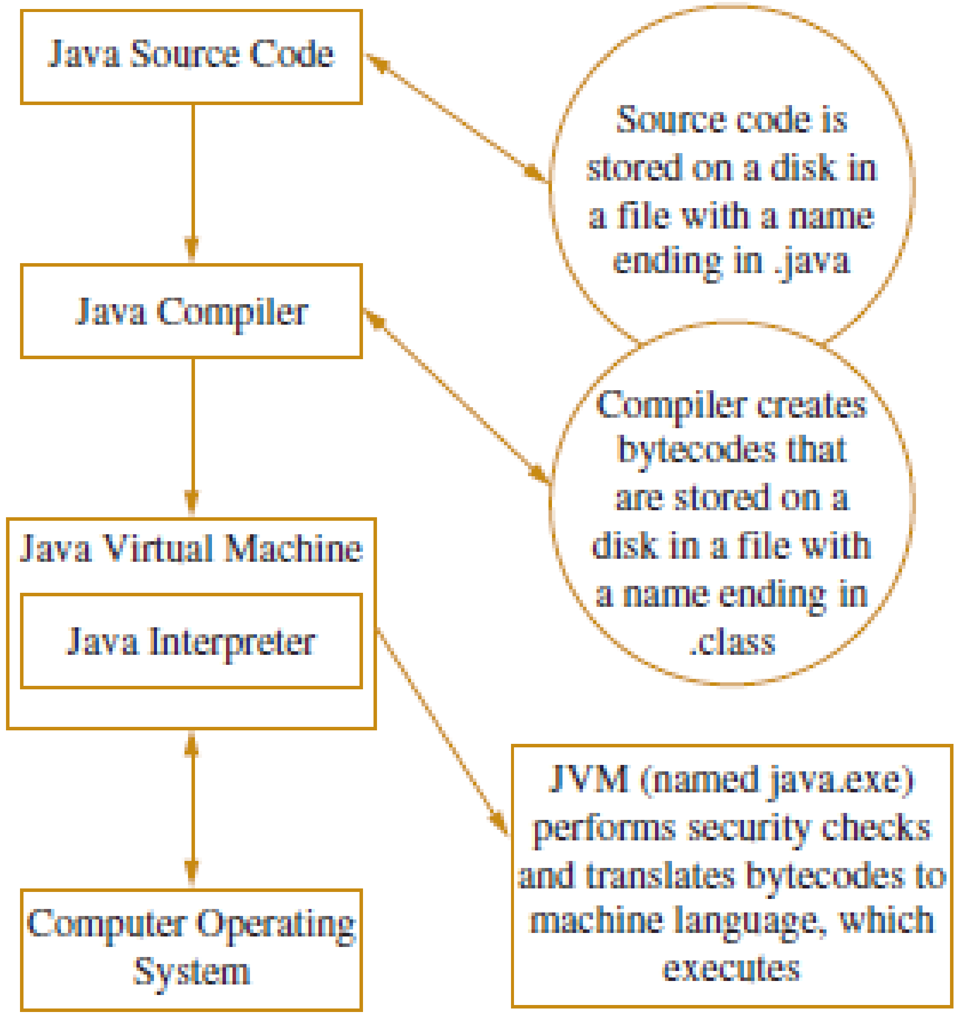
Comments

```
// Demonstrating comments
/* This shows
   that these comments
   don't matter */
System.out.println("Hello"); // This line executes
  // up to where the comment started
/* Everything but the println()
    is a comment */
```

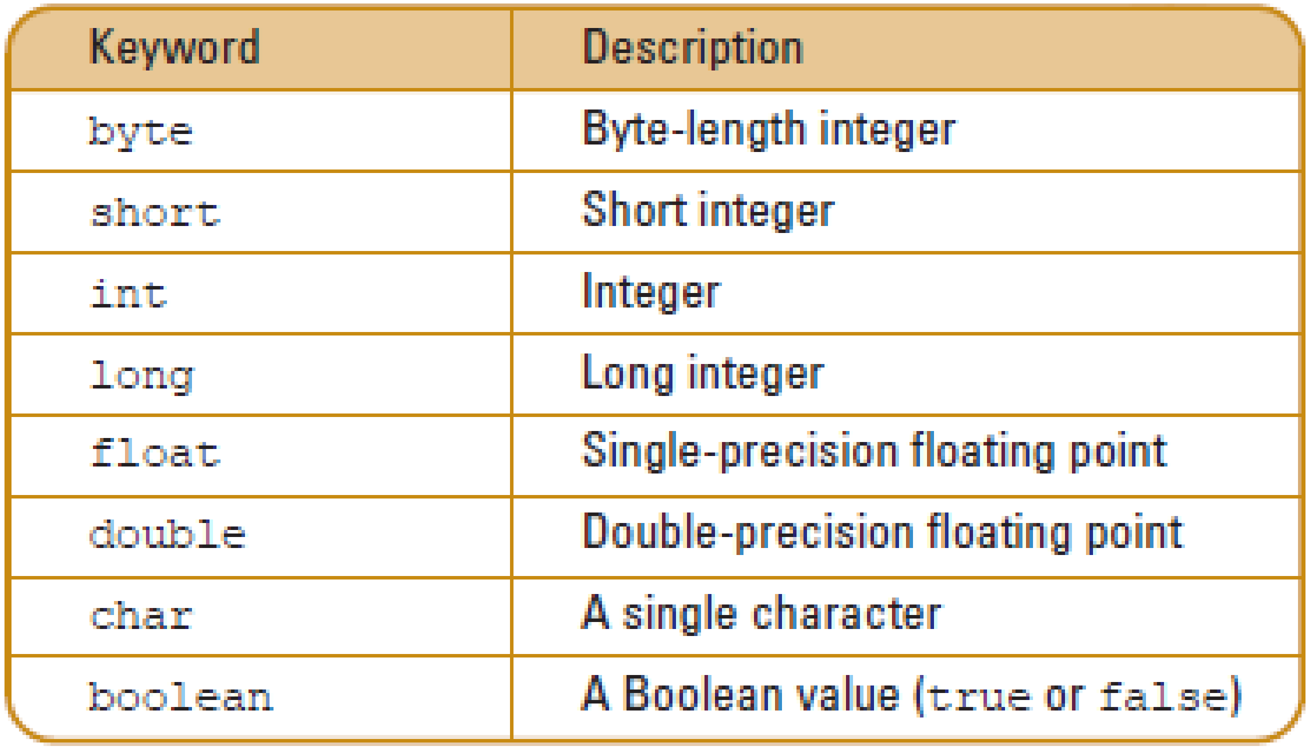
Displaying a message

```java
import javax.swing.JOptionPane;
public class FirstDialog
{
   public static void main(String[] args)
   {
      JOptionPane.showMessageDialog(null, "First Java dialog");
   }
}
```

JVM


Java Source Code
Java Compiler
Java Virtual Machine
Java Interpreter
Computer Operating System
Source code is stored on a disk in a file with a name ending in .java
Compiler creates bytecodes that are stored on a disk in a file with a name ending in .class
JVM (named java.exe) performs security checks and translates bytecodes to machine language, which executes

Basic data types

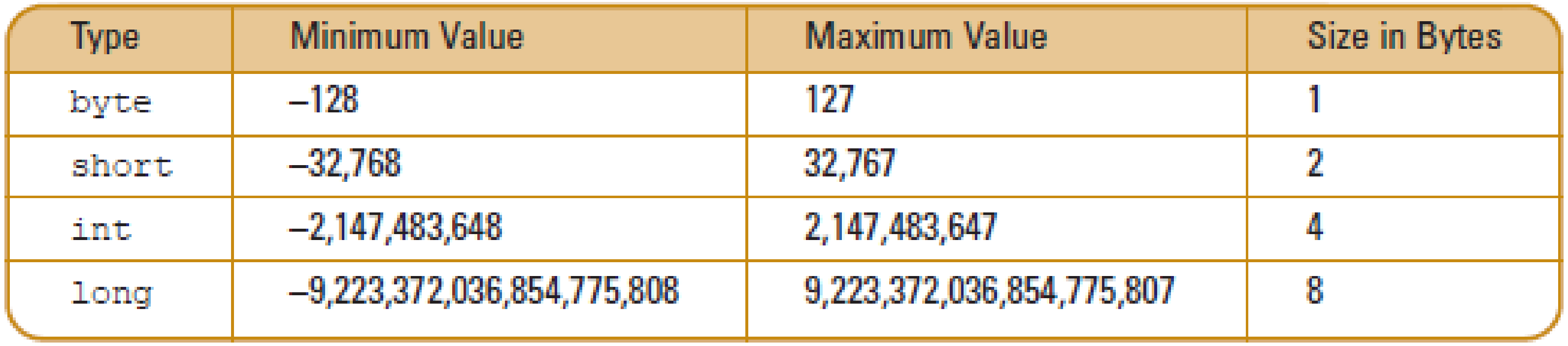


Keyword	Description
`byte`	Byte-length integer
`short`	Short integer
`int`	Integer
`long`	Long integer
`float`	Single-precision floating point
`double`	Double-precision floating point
`char`	A single character
`boolean`	A Boolean value (`true` or `false`)

Basic Types

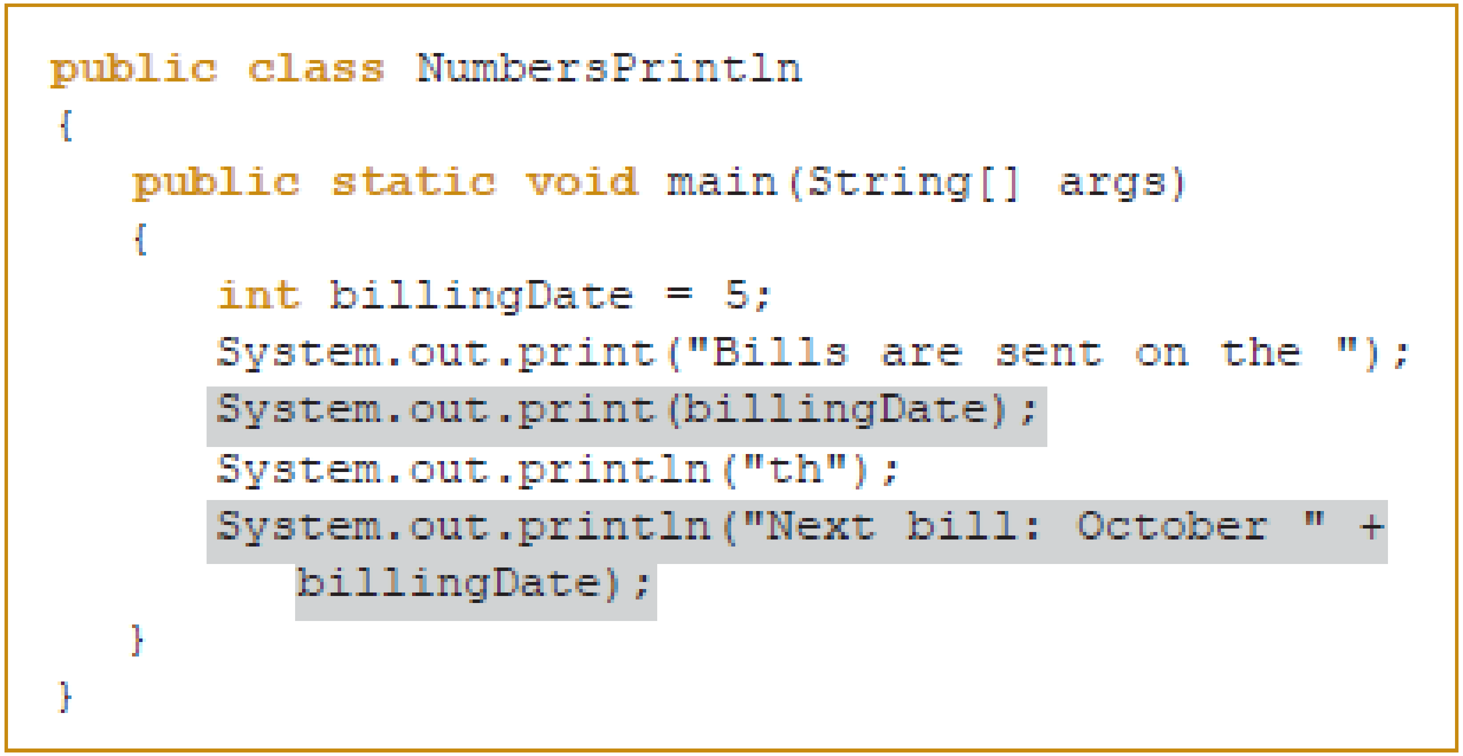
```
byte v1 = 126;   // 8 bits. -128 to +127

short v2 = -32000;   //16 bits. -32,768 to +32,767

int v3= 1234567890;   // 32 bits. -2,147,483,648 to +2,147,483,647

long v5 = 123456789L;   //64 bits.

float v4 = 35.6F;   // 32 bits.

double v6 = 123456789.12345D;   //64 bits.

char v7 ='C';   //16 bits.

boolean v8 = true; // or false   // 1 bit.
```

Integer



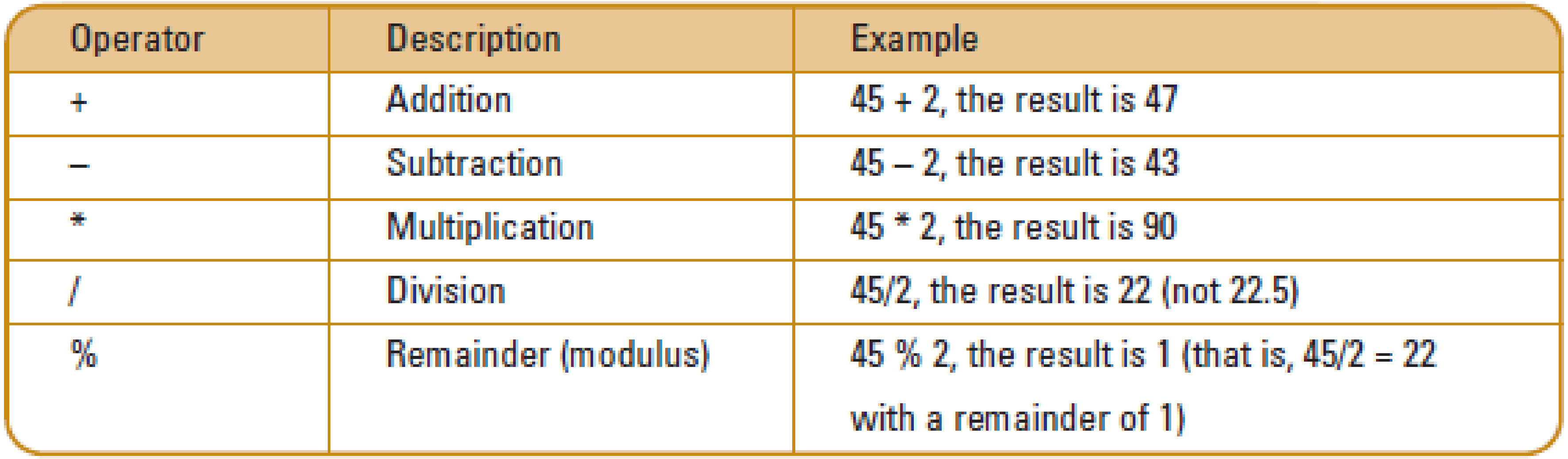
Type	Minimum Value	Maximum Value	Size in Bytes
`byte`	–128	127	1
`short`	–32,768	32,767	2
`int`	–2,147,483,648	2,147,483,647	4
`long`	–9,223,372,036,854,775,808	9,223,372,036,854,775,807	8

Integer



```java
public class NumbersPrintln
{
   public static void main(String[] args)
   {
      int billingDate = 5;
      System.out.print("Bills are sent on the ");
      System.out.print(billingDate);
      System.out.println("th");
      System.out.println("Next bill: October " +
         billingDate);
   }
}
```

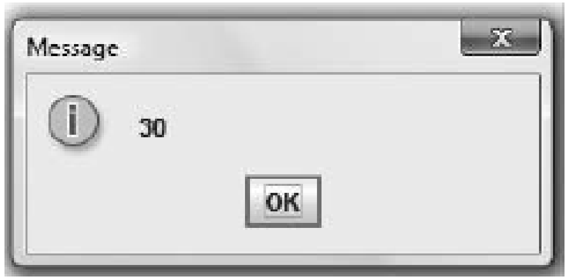
Arithmetic Operators



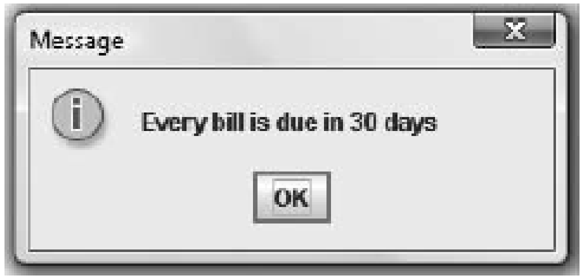
Operator	Description	Example
+	Addition	45 + 2, the result is 47
–	Subtraction	45 – 2, the result is 43
*	Multiplication	45 * 2, the result is 90
/	Division	45/2, the result is 22 (not 22.5)
%	Remainder (modulus)	45 % 2, the result is 1 (that is, 45/2 = 22 with a remainder of 1)

Integer

```
import javax.swing.JOptionPane;
public class NumbersDialog
{
    public static void main(String[] args)
    {
        int creditDays = 30;
        JOptionPane.showMessageDialog(null,"" + creditDays);
        JOptionPane.showMessageDialog
            (null, "Every bill is due in " + creditDays + " days");
    }
}
```



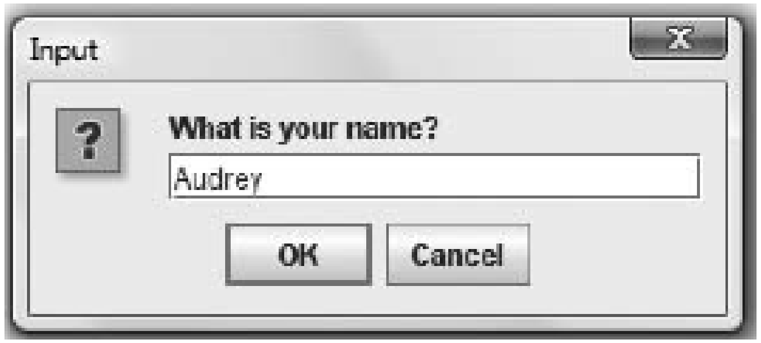
30

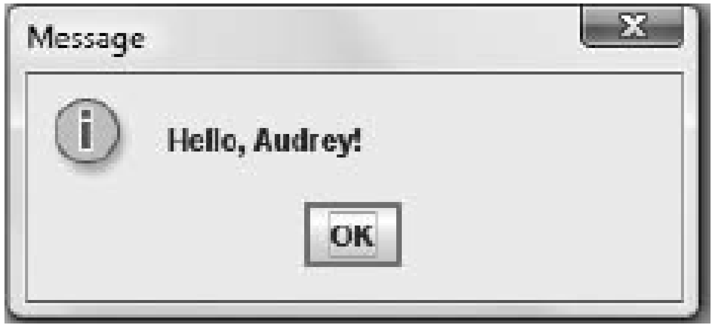


Every bill is due in 30 days

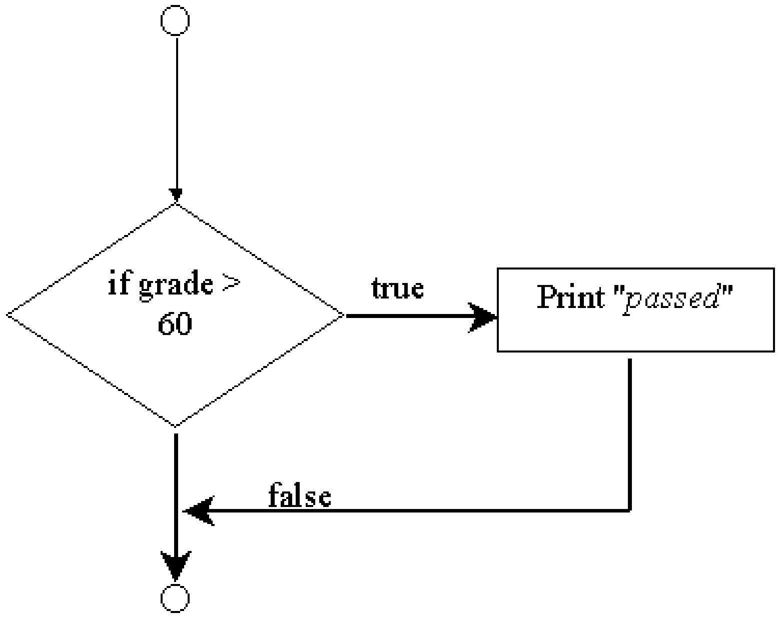
Reading a String from the user

```
import javax.swing.JOptionPane;
public class HelloNameDialog
{
   public static void main(String[] args)
   {
      String result;
      result = JOptionPane.showInputDialog(null, "What is your name?");
      JOptionPane.showMessageDialog(null, "Hello, " + result + "!");
   }
}
```





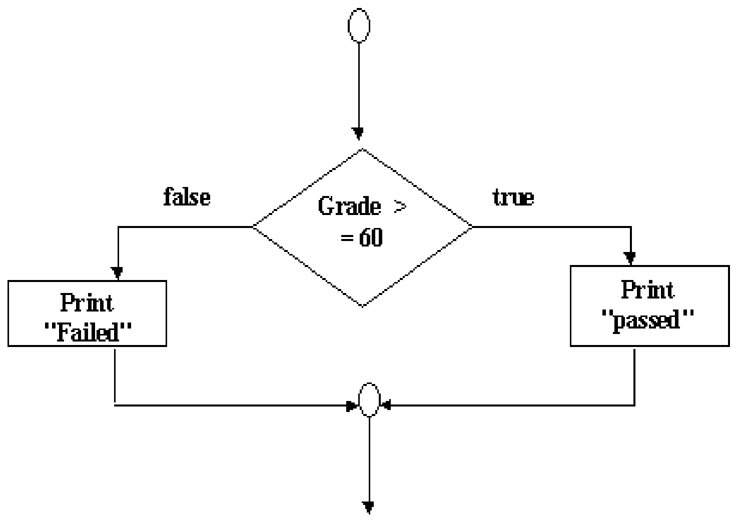
Control Structures: if



```
public class MyFirstProgram
{
 public static void main(String args[])
 {
   int grade;
   grade = 80;
   if(grade>60)
        System.out.println("passed");
 }
}
```

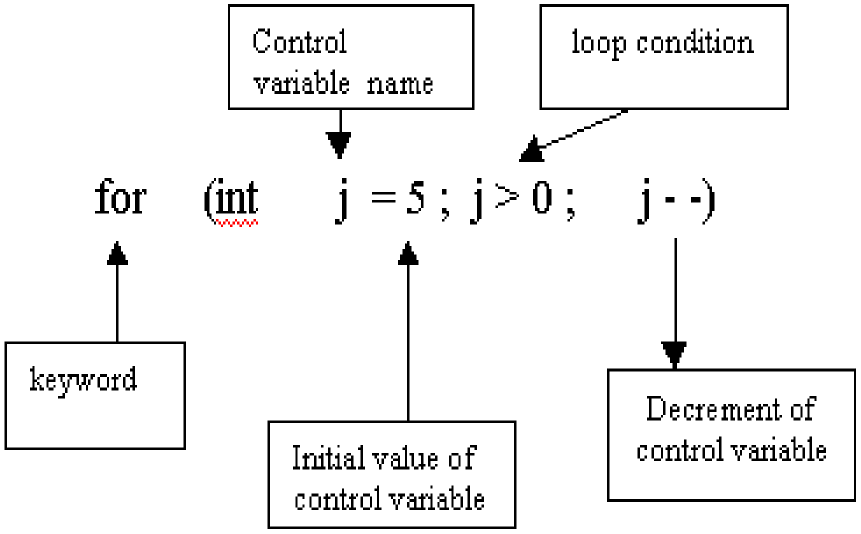
Control Structures: if .. else

```
public class MyFirstProgram
{
  public static void main(String args[])
 {
    int grade;
    grade = 40 ;
    if(grade>60)
        System.out.println("passed");
    else
        System.out.println("failed");
  }
}
```



Control Structures: for

```
public MyApplication{
public static void main(String args[ ])
{
        for(int j=5; j > 0; j--)
        {
          System.out.println("Hello");
        }
        System.out.println(" Good Bye ");
  }
}
```



Control variable name
loop condition
for (int j = 5 ; j > 0 ; j - -)
keyword
Initial value of control variable
Decrement of control variable

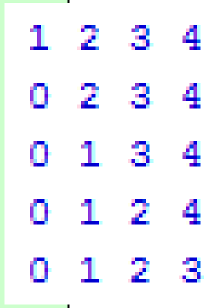
Control Structures: breaking a loop

```
public MyApplication{
public static void main(String args[ ]) {
        for(int i = 0; i<5; i++) {
        for(int j=0; j <5; j++) {
           if(i==j)
              break;
          else
              System.out.print(" "+j);
        }
        System.out.println("");
         }
 }
}
```

```
0
0 1
0 1 2
0 1 2 3
```

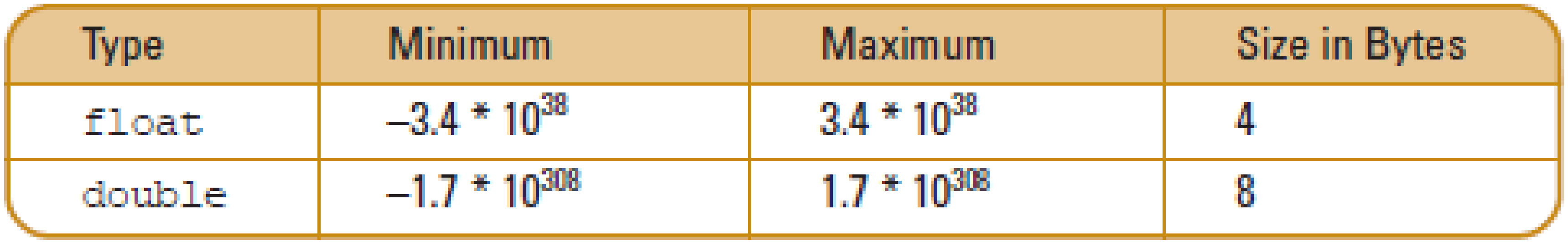
Control Structures: Skipping an iteration

```
public MyApplication{
public static void main(String args[ ]) {
        for(int i = 0; i<5; i++) {
        for(int j=0; j <5; j++) {
           if(i==j)
              continue;
          else
              System.out.print(" "+j);
        }
        System.out.prntln("");
         }
  }
}
```



```
1  2  3  4
0  2  3  4
0  1  3  4
0  1  2  4
0  1  2  3
```

Decimal Numbers

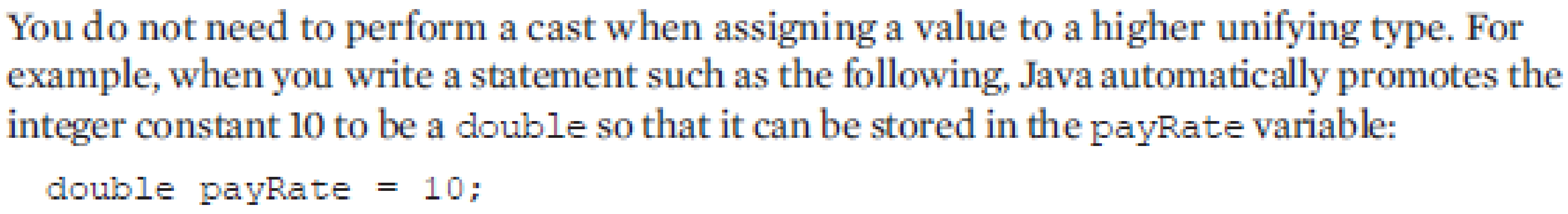


Type	Minimum	Maximum	Size in Bytes
`float`	$-3.4 * 10^{38}$	$3.4 * 10^{38}$	4
`double`	$-1.7 * 10^{308}$	$1.7 * 10^{308}$	8

Type Conversion

```
int hoursWorked = 37;
double payRate = 6.73;
int grossPay = hoursWorked * payRate;  //ERROR

double bankBalance = 189.66;
float weeklyBudget = (float) bankBalance / 4;

float myMoney = 47.82f;
int dollars = (int) myMoney;
   // dollars is 47, the integer part of myMoney
```

Decimal Numbers


You do not need to perform a cast when assigning a value to a higher unifying type. For example, when you write a statement such as the following, Java automatically promotes the integer constant 10 to be a `double` so that it can be stored in the `payRate` variable:

```
double payRate = 10;
```

Note the all the arithmetic operators (except %), works with decimal number to produce decimal numbers.

```
int x = 2;
float y = 4.6f;
float z = y/x;      //z = 2.3f;
```

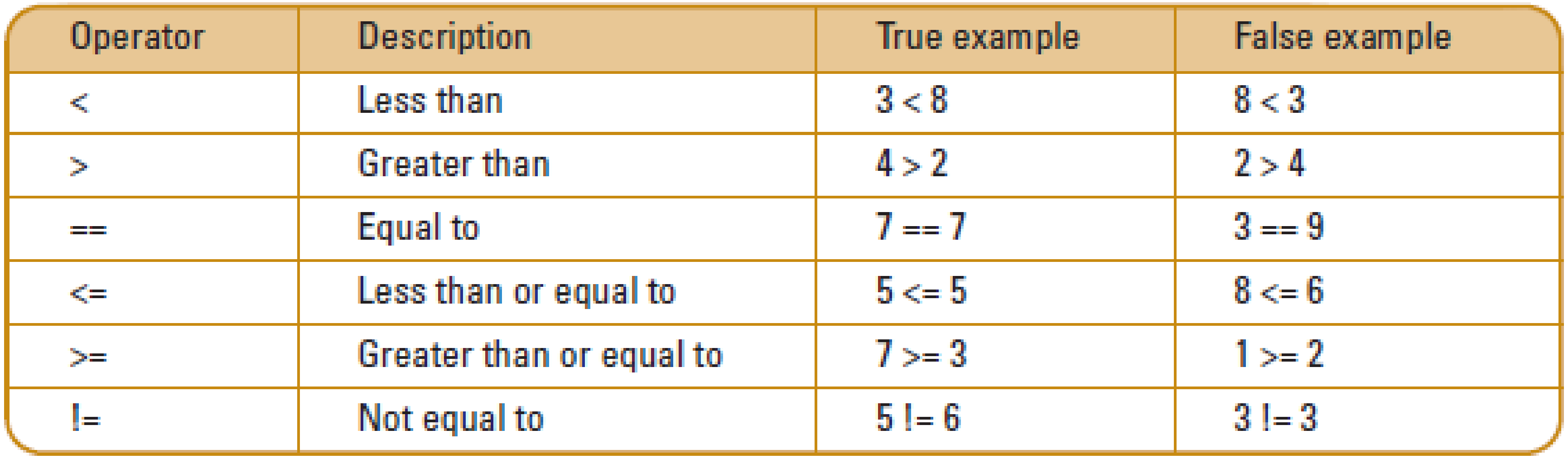
char

```
char myMiddleInitial = 'M';
char myGradeInChemistry = 'A';
char aStar = '*';
```

Escape Sequence	Description
\b	Backspace; moves the cursor one space to the left
\t	Tab; moves the cursor to the next tab stop
\n	Newline or linefeed; moves the cursor to the beginning of the next line
\r	Carriage return; moves the cursor to the beginning of the current line
\"	Double quotation mark; prints a double quotation mark
\'	Single quotation mark; prints a single quotation mark
\\	Backslash; prints a backslash character

```
public class HelloThereNewLine
{
   public static void main(String[] args)
   {
      System.out.println("Hello\nthere");
   }
}
```

Logical Operators



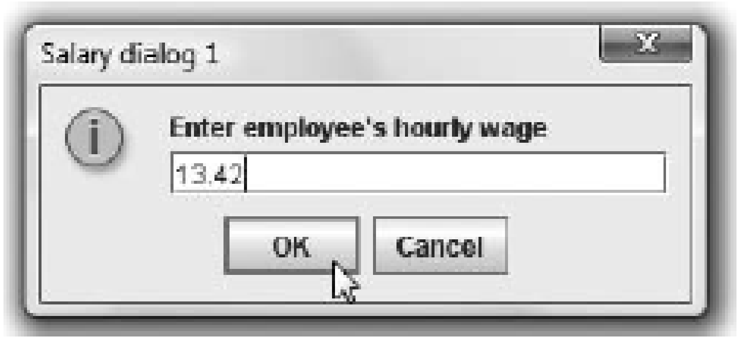
Operator	Description	True example	False example
<	Less than	3 < 8	8 < 3
>	Greater than	4 > 2	2 > 4
==	Equal to	7 == 7	3 == 9
<=	Less than or equal to	5 <= 5	8 <= 6
>=	Greater than or equal to	7 >= 3	1 >= 2
!=	Not equal to	5 != 6	3 != 3

Reading Numbers from the user

```
import javax.swing.JOptionPane;
public class SalaryDialog
{
   public static void main(String[] args)
   {
       String wageString, dependentsString;
       double wage, weeklyPay;
       int dependents;
       final double HOURS_IN_WEEK = 37.5;
       wageString = JOptionPane.showInputDialog(null,
          "Enter employee's hourly wage", "Salary dialog 1",
          JOptionPane.INFORMATION_MESSAGE);
       weeklyPay = Double.parseDouble(wageString) *HOURS_IN_WEEK;
       dependentsString = JOptionPane.showInputDialog(null,
          "How many dependents?", "Salary dialog 2",
          JOptionPane.QUESTION_MESSAGE);
       dependents = Integer.parseInt(dependentsString);
       JOptionPane.showMessageDialog(null, "Weekly salary is $" +
          weeklyPay + "\nDeductions will be made for " +
          dependents + " dependents");
   }
}
```

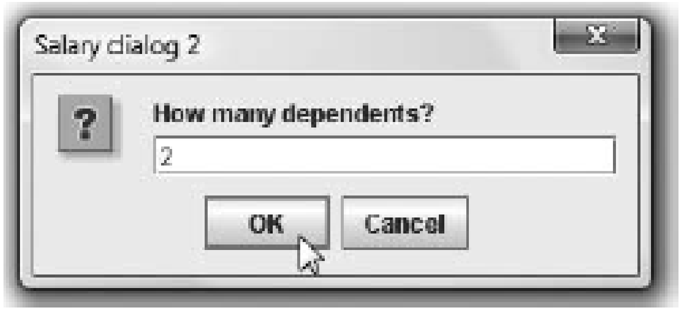
Reading Numbers from the user

```
wageString = JOptionPane.showInputDialog(null,
   "Enter employee's hourly wage", "Salary dialog 1",
   JOptionPane.INFORMATION_MESSAGE);
```


Salary dialog 1

Enter employee's hourly wage

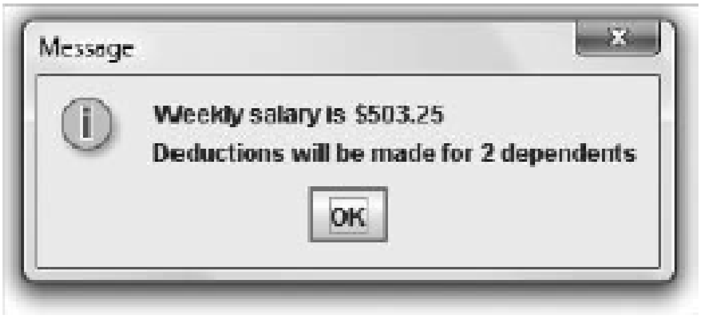
13.42


Salary dialog 2

How many dependents?

2

```
dependentsString = JOptionPane.showInputDialog(null,
   "How many dependents?", "Salary dialog 2",
   JOptionPane.QUESTION_MESSAGE);
```

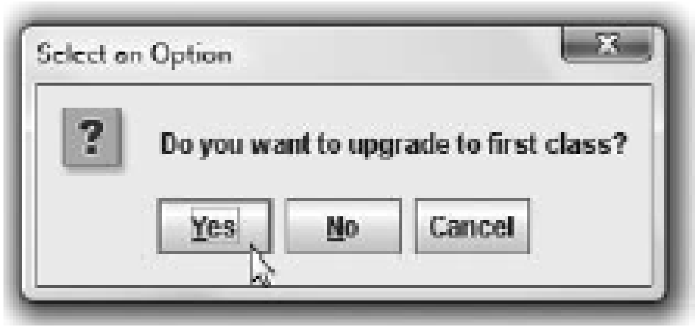

Message

Weekly salary is $503.25
Deductions will be made for 2 dependents

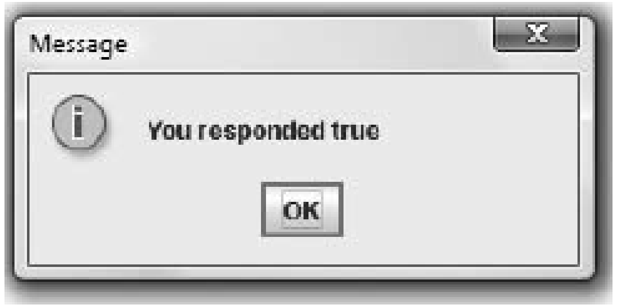
```
JOptionPane.showMessageDialog(null, "Weekly salary is $" +
   weeklyPay + "\nDeductions will be made for " +
   dependents + " dependents");
```

Confirmation Messages

```
import javax.swing.JOptionPane;
public class AirlineDialog
{
   public static void main(String[] args)
   {
      int selection;
      boolean isYes;
      selection = JOptionPane.showConfirmDialog(null,
         "Do you want to upgrade to first class?");
      isYes = (selection == JOptionPane.YES_OPTION);
      JOptionPane.showMessageDialog(null,
         "You responded " + isYes);
   }
}
```



Do you want to upgrade to first class?



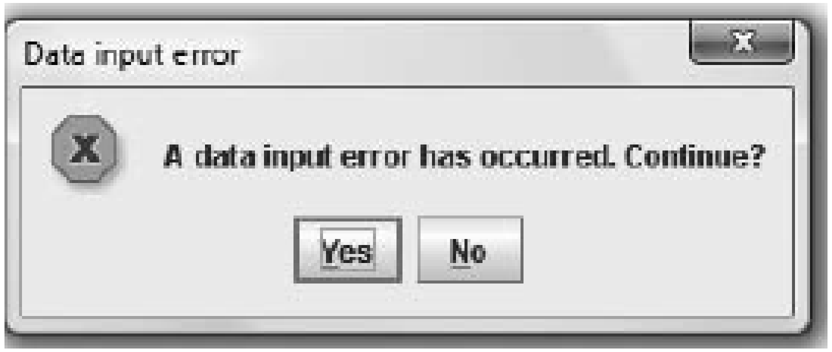
You responded true

Confirmation Messages

You can also create a confirm dialog box with five arguments, as follows:

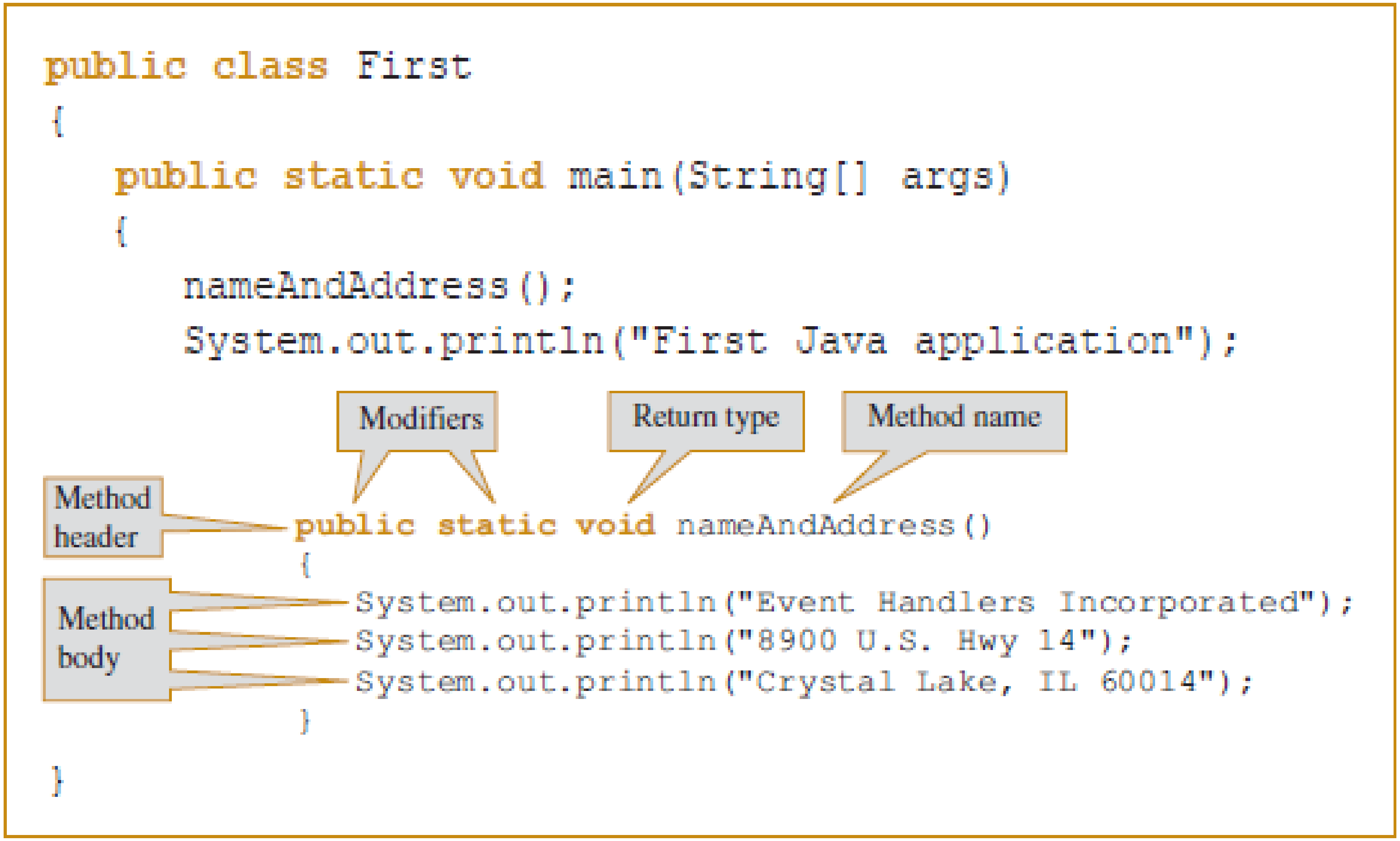
- The parent component, which can be null
- The prompt message
- The title to be displayed in the title bar
- An integer that indicates which option button will be shown (It should be one of the class variables YES_NO_CANCEL_OPTION or YES_NO_OPTION.)
- An integer that describes the kind of dialog box (It should be one of the class variables ERROR_MESSAGE, INFORMATION_MESSAGE, PLAIN_MESSAGE, QUESTION_MESSAGE, or WARNING_MESSAGE.)

```
JOptionPane.showConfirmDialog(null,
     "A data input error has occurred. Continue?",
     "Data input error", JOptionPane.YES_NO_OPTION,
      JOptionPane.ERROR_MESSAGE);
```


Data input error

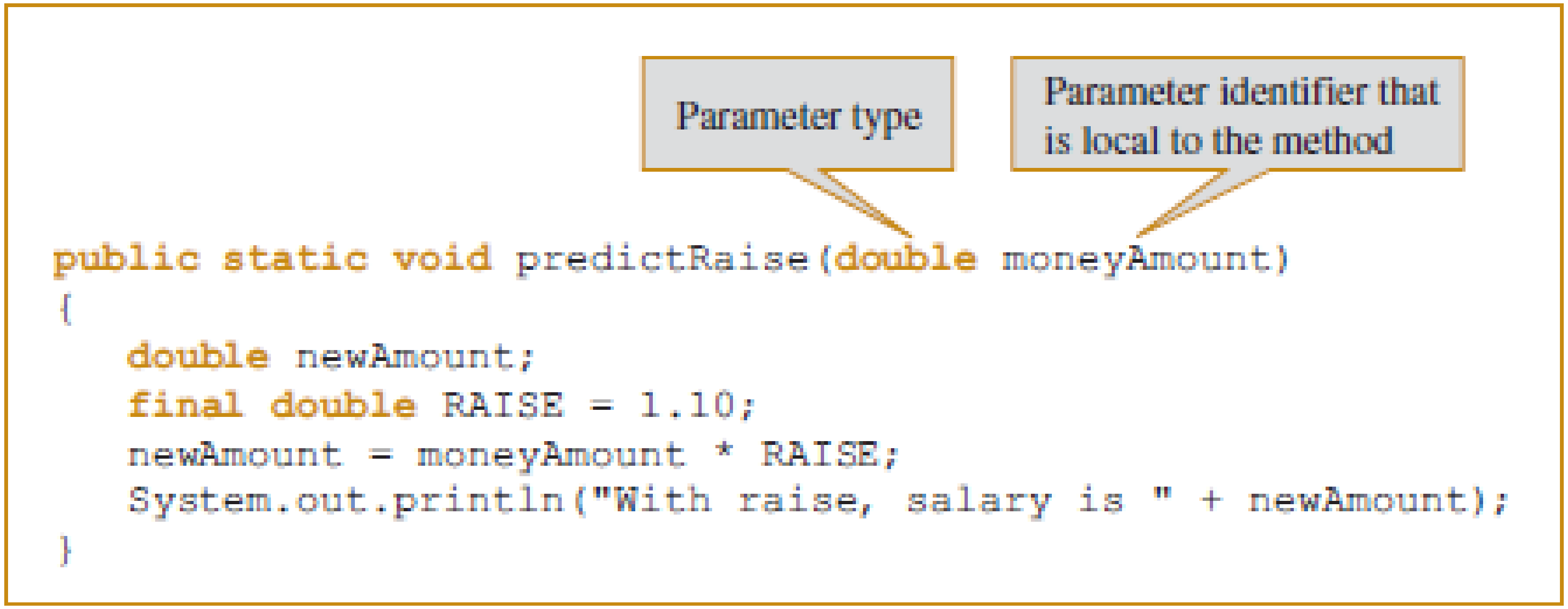
A data input error has occurred. Continue?

First Method



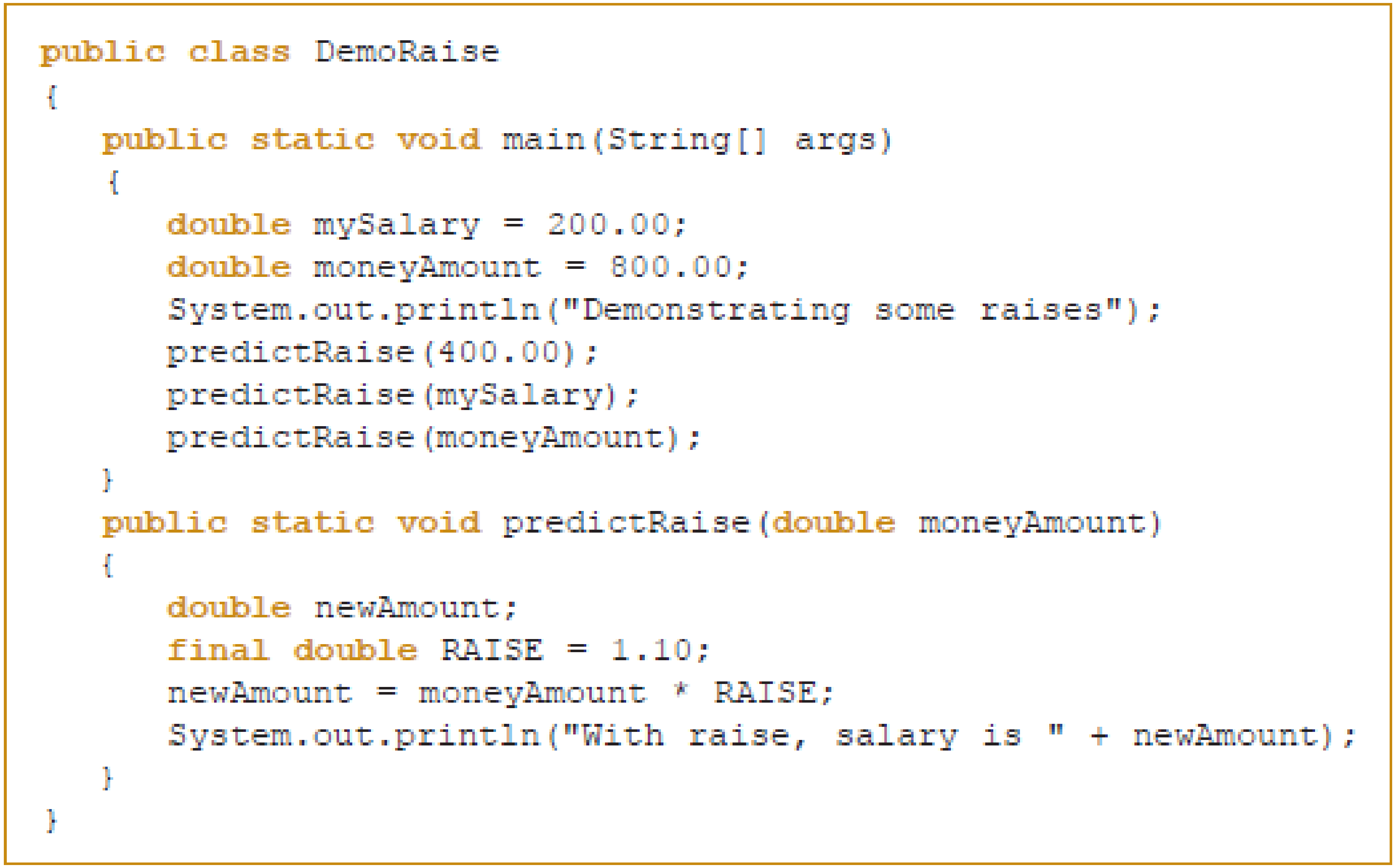
```
public class First
{
    public static void main(String[] args)
    {
        nameAndAddress();
        System.out.println("First Java application");

    public static void nameAndAddress()
    {
        System.out.println("Event Handlers Incorporated");
        System.out.println("8900 U.S. Hwy 14");
        System.out.println("Crystal Lake, IL 60014");
    }
}
```

A method with a single parameter



```
public static void predictRaise(double moneyAmount)
{
    double newAmount;
    final double RAISE = 1.10;
    newAmount = moneyAmount * RAISE;
    System.out.println("With raise, salary is " + newAmount);
}
```

A method with a single parameter

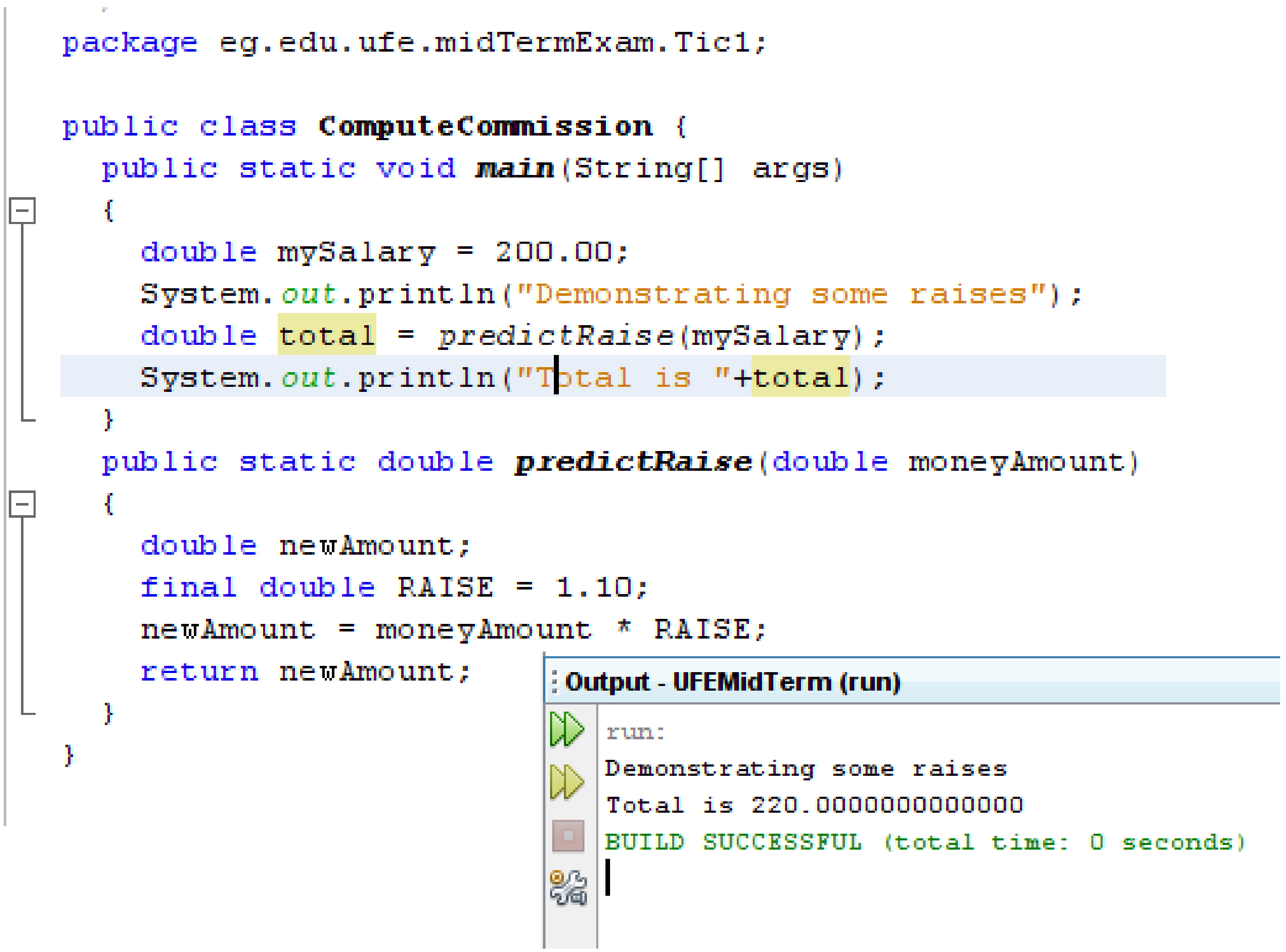


```java
public class DemoRaise
{
   public static void main(String[] args)
   {
      double mySalary = 200.00;
      double moneyAmount = 800.00;
      System.out.println("Demonstrating some raises");
      predictRaise(400.00);
      predictRaise(mySalary);
      predictRaise(moneyAmount);
   }
   public static void predictRaise(double moneyAmount)
   {
      double newAmount;
      final double RAISE = 1.10;
      newAmount = moneyAmount * RAISE;
      System.out.println("With raise, salary is " + newAmount);
   }
}
```

A method with multiple parameters

```
public class ComputeCommission
{
   public static void main(String[] args)
   {
      char vType = 'S';
      int value = 23000;
      double commRate = 0.08;
      computeCommission(value, commRate, vType);
      computeCommission(40000, 0.10, 'L');
   }
   public static void computeCommission(int value,
      double rate, char vehicle)
   {
      double commission;
      commission = value * rate;
      System.out.println("\nThe " + vehicle +
          " type vehicle is worth $" + value);
      System.out.println("With " + (rate * 100) +
          "% commission rate, the commission is $" +
          commission);
   }
}
```

A method that returns a parameter



```
package eg.edu.ufe.midTermExam.Tic1;

public class ComputeCommission {
  public static void main(String[] args)
  {
    double mySalary = 200.00;
    System.out.println("Demonstrating some raises");
    double total = predictRaise(mySalary);
    System.out.println("Total is "+total);
  }
  public static double predictRaise(double moneyAmount)
  {
    double newAmount;
    final double RAISE = 1.10;
    newAmount = moneyAmount * RAISE;
    return newAmount;
  }
}
```

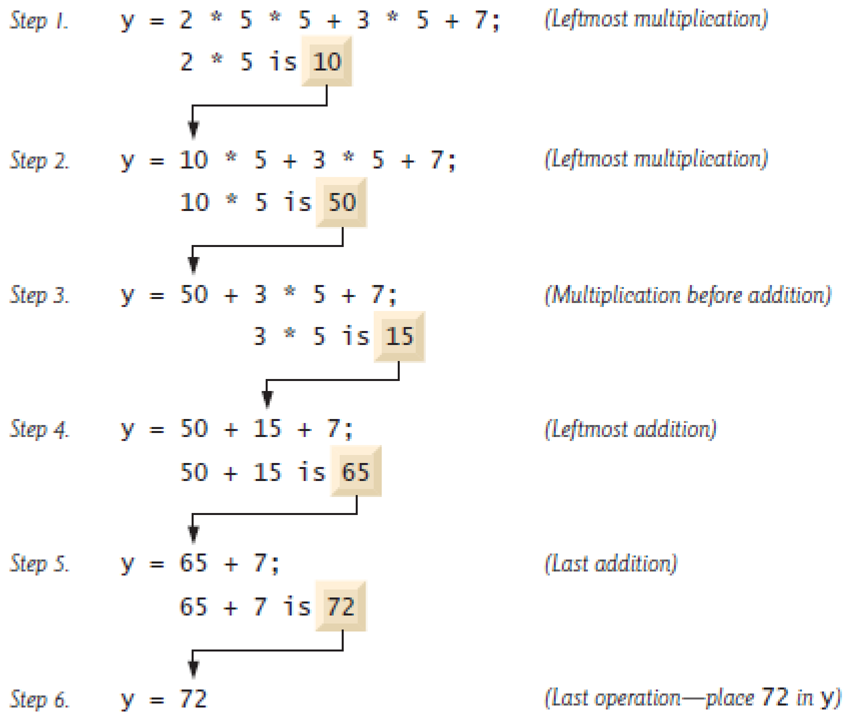
Output - UFEMidTerm (run)

```
run:
Demonstrating some raises
Total is 220.0000000000000
BUILD SUCCESSFUL (total time: 0 seconds)
```

Arithmetic Precedence

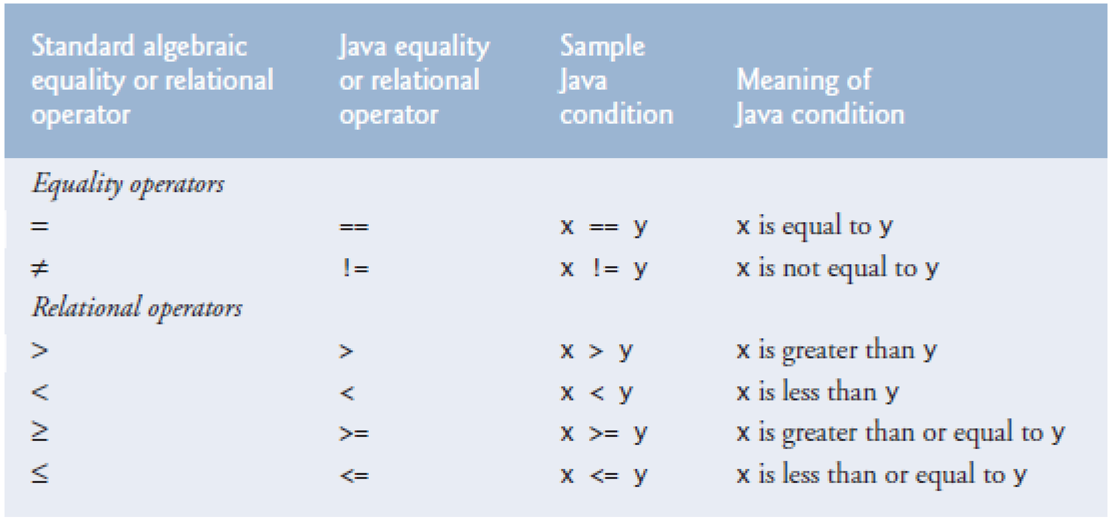
Operator(s)	Operation(s)	Order of evaluation (precedence)
* / %	Multiplication Division Remainder	Evaluated first. If there are several operators of this type, they're evaluated from left to right.
+ -	Addition Subtraction	Evaluated next. If there are several operators of this type, they're evaluated from left to right.
=	Assignment	Evaluated last.

Precedence of arithmetic operators.



```
Step 1.   y = 2 * 5 * 5 + 3 * 5 + 7;      (Leftmost multiplication)
              2 * 5 is 10
Step 2.   y = 10 * 5 + 3 * 5 + 7;         (Leftmost multiplication)
              10 * 5 is 50
Step 3.   y = 50 + 3 * 5 + 7;             (Multiplication before addition)
                   3 * 5 is 15
Step 4.   y = 50 + 15 + 7;                (Leftmost addition)
              50 + 15 is 65
Step 5.   y = 65 + 7;                     (Last addition)
              65 + 7 is 72
Step 6.   y = 72                          (Last operation—place 72 in y)
```

Relational Operators



Standard algebraic equality or relational operator	Java equality or relational operator	Sample Java condition	Meaning of Java condition
Equality operators			
=	`==`	`x == y`	`x` is equal to `y`
≠	`!=`	`x != y`	`x` is not equal to `y`
Relational operators			
>	`>`	`x > y`	`x` is greater than `y`
<	`<`	`x < y`	`x` is less than `y`
≥	`>=`	`x >= y`	`x` is greater than or equal to `y`
≤	`<=`	`x <= y`	`x` is less than or equal to `y`

Equality and relational operators.

Control Structures: switch

```
public class MyFirstProgram {
 public static void main(String args[])
{
        int i =4;
        switch(i)
        {
          case 1: System.out.println("1");
                  break;
          case 2: System.out.println( "2");
                  break;
          default: System.out.println("another number");
                  break;
      }
 }
}
```

Control Structures: switch

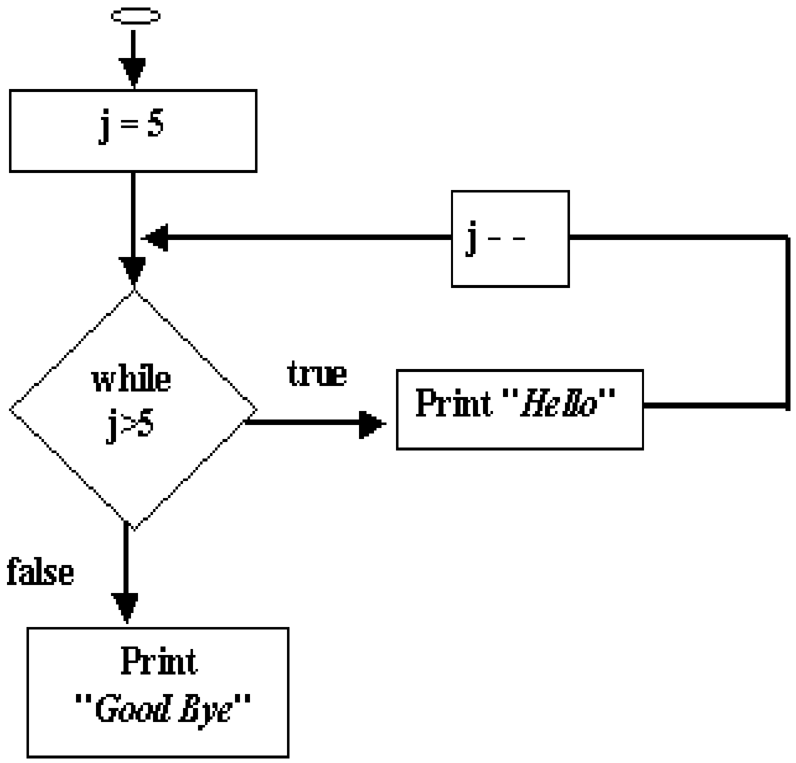
```
public class MyFirstProgram {
  public static void main(String args[])  {
        char c='b';
        switch(c)
        {
          case 'a': System.out.println("a");
                    break;
          case 'b': System.out.println("b");
                    break;
          default: System.out.println("another character");
                    break;
        }
  }
}
```

Control Structures: switch

```
public class SwitchString {
   public static void main(String args[])  {
    String test="CS";
    switch(test)
    {
     case "CS": System.out.println("Computer Science");
           break;
     case "CE": System.out.println("Computer Engineering");
           break;
     default: System.out.println("Other");
           break;
    }
 }
```

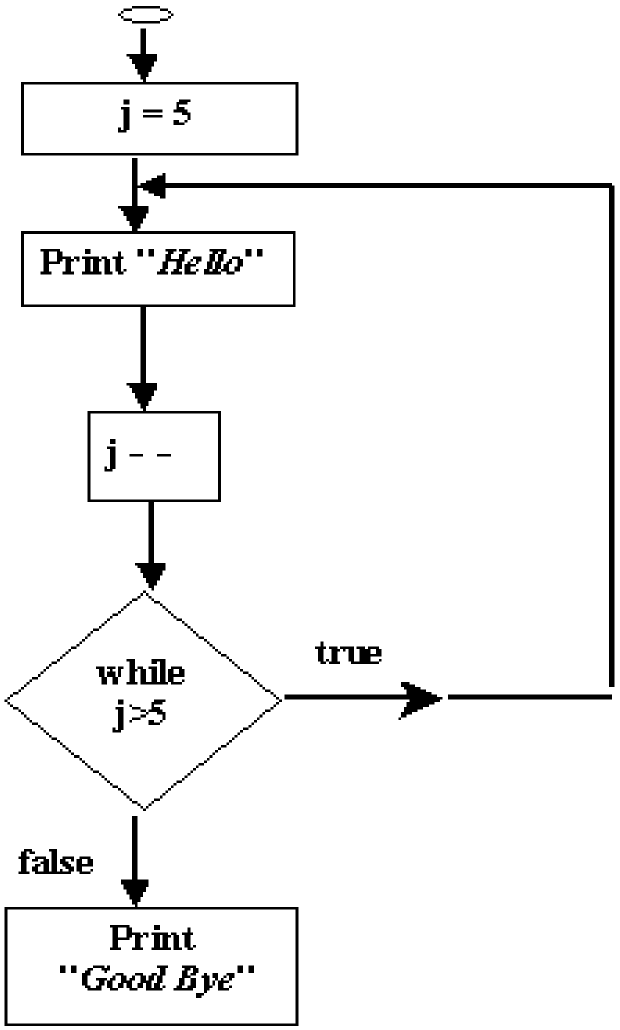
Control Structures: while

```
public MyApplication{
public static void main(String args[ ])
{
        int j=5;
        while(j > 0)
        {
           System.out.println("Hello");
           j--;
         }
        System.out.println(" Good Bye ");
 }
}
```



Control Structures: do … while

```
public MyApplication{
public static void main(String args[ ])
{
  char c=' ';
  do
   {
    System.out.print("Java is cool ");
    System.out.println("do you want to see"+
                      " the message again (Y/N) ");
      c = System.in.read();
    } while((c =='Y')||(c =='y'));
}
}
```


j = 5
Print "Hello"
j - -
while j>5
true
false
Print "Good Bye"