2.1 Character Strings

Every character string is an object in Java, defined by the `String` class**.** Every string literal, delimited by double quotation marks, represents a `String` object**.** The *string concatenation operator* (+) is used to append one string to the end of another. It can also be used to append a number to a string. A string literal cannot be broken across two lines in a program.

```
System.out.println ("Here are examples to clarify  "
                        + "string concatenation:");

System.out.println ();

// A string can contain numeric digits
System.out.println ("Letters in the Hawaiian alphabet: 12");

// A numeric value can be concatenated to a string
System.out.println ("Speed of ketchup: " + 40 + " km per year");
```

Output:

The plus operator (+) is also used for arithmetic addition The function that the + operator performs depends on the type of the information on which it operates. If both operands are strings, or if one is a string and one is a number, it performs string concatenation. If both operands are numeric, it adds them. The + operator is evaluated left to right. Parentheses can be used to force the operation order.

```
System.out.println ("24 and 40 concatenated: " + 24 + 40);
System.out.println ("24 and 40 added: " + (24 + 40));
System.out.println (24 + 40 + " = 24 and 40 added");
```

Output:

println vs print

`System.out.println` prints the character string and returns the cursor to the next line.

`System.out.print` prints the character string and leaves the cursor on the same line after the last character printed. Write the output from the following statements:

```
System.out.println ("APCS is fun!");
System.out.print ("Three…");
System.out.print ("Two…");
System.out.print ("One…");
System.out.println ("Liftoff!");
```

Output:

Escape Sequences

Some characters have special meaning in Java. A double quote represents the beginning or ending of a string of characters. If you want to print out a double quote, you need to use a backslash character (called an escape sequence) to tell the compiler the next character will be treated in a special way.

```
System.out.println ("I said \"Hello\" to you.");
```

Output:

Escape Sequence	Meaning
\n	newline
\"	double quote
\'	single quote
\\	backslash
\t	tab (not on AP)

```
System.out.println ("Roses are red,\n\tViolets are blue,\n" +
"Sugar is sweet,\n\tBut I have \"commitment issues\",\n\t" +
"So I'd rather just be friends.");
```

Output:

What `System.out.println` statement would you use to print out the following character sequence?

\\//"\\\//"\\//

2.2 Variables

A *variable* is a name for a location in memory. A variable must be *declared* by specifying the variable's name and the type of information that it will hold. A variable is like a cup or container. It holds something. Multiple variables can be created in one declaration.

```
int total;

int temp, count, result;
```

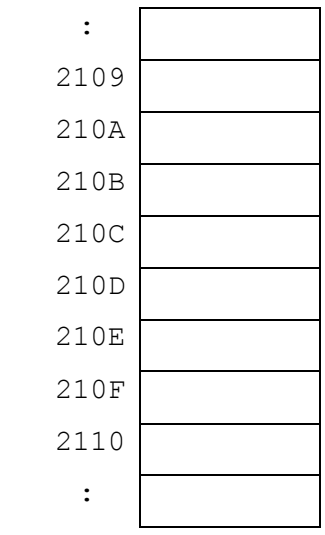
A variable can be given an initial value in the declaration. If you don't explicitly assign a value to an instance variable, the instance variable is given a default value. `int` and `double` types get zero (0), `boolean` gets false and object reference types get `null`.

```
int sum = 0;
int base = 32, max = 149;
```

An *assignment statement* changes the value of a variable. The assignment operator is the = sign.

```
total = 55;
```

The expression on the right is evaluated and the result is stored in the variable on the left. The value that was in total is overwritten.



Constants

A *constant* is an identifier that is similar to a variable except that it holds one value while the program is active. The compiler will issue an error if you try to change the value of a constant during execution. In Java, we use the *final* modifier to declare a constant. It is the convention to declare constants with all uppercase letters and use and underscore between words. Class constants must be initialized with an initializer:

```
final int MIN_HEIGHT = 69;
```

Primitive Data Types

The three primitive data types we will use are `int`, `double` and `boolean`. An `int` takes up 32 bits of storage and has a minimum value of -2,147,483,648 and a maximum value of 2,147,483,647. `double` takes up 64 bits of storage and have minimum and maximum value of +/- 1.7 x 10^{308} with 15 significant digits. `boolean` variables have a value of `true` or `false`.

- What is the difference between a variable and a constant?

- Explain what each of the lines below does. Be sure to indicate how each is different from the others.
  a. `int x;`

  b. `int x = 3;`

  c. `x = 3;`

2.3 Expressions

An *expression* is a combination of one or more operands and their operators. *Arithmetic expressions* compute numeric results and make use of the arithmetic operators:

Addition	+
Subtraction	-
Multiplication	*
Division	/
Remainder	% (mod)

Syntax for declarations

type identifier = expression;

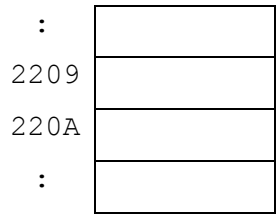
where:
type is the type of the variable
identifier is the name of the variable
expression specifies the initial value

If both operands to the division operator (/) are integers, the result is an integer (the fractional part is discarded). This behavior is called "truncation towards 0".

```
14 / 3          -14 / 3          8 / 12
```

If either or both operands associated with an arithmetic operator are floating point, the result is a floating point.

```
double result;

result = (2 + 4 + 10) / 3;

result = (2 + 4 + 10) / 3.0;
```

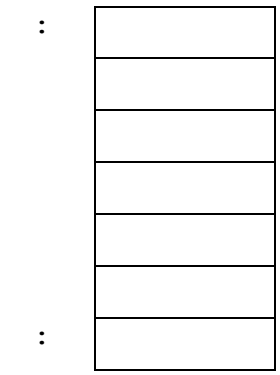


The remainder operator (%) returns the remainder after dividing the second operand into the first. For example, `9 % 4 = 1`, since `4` goes into `9` twice, with `1` left over.

```
14 % 3 =                    8 % 12 =                    0 % 4 =

1 % 4 =                     4 % 4 =                     20 % 4 =
```

Operators can be combined into complex expressions:

```
double result;
int total = 40, count = 10, max = 2, offset = 3;
result  =  total + count / max - offset;
```



Operators have a well-defined precedence which determines the order in which they are evaluated. What is the order of evaluation in the following expressions?

```
a + b + c + d + e                a + b * c - d / e


a / (b + c) - d % e              a / (b * (c + (d - e)))
```

Data Conversions

Sometimes it is convenient to convert data from one type to another. For example, we may want to treat an integer as a floating point value during a computation. Conversions must be handled carefully to avoid losing information. *Widening conversions* are safest because they usually do not lose information (`int` to `double`). *Narrowing conversions* can lose information (`double` to `int`).

In Java, data conversions can occur in three ways:

- assignment conversion
- arithmetic promotion
- casting

Assignment conversion occurs when a value of one type is assigned to a variable of another. Only widening conversions can happen via assignment.

```
int x = 4;
double y;
y = x;
x = y;
```

Arithmetic promotion happens automatically when operators in expressions convert their operands.

```
double result, sum = 2.0;
int count = 4;
result = sum / count;
```

Casting is the most powerful, and dangerous, technique for conversion. Both widening and narrowing conversions can be accomplished by explicitly casting a value. To cast, the *type* is put in parenthesis `()` in front of the value being converted. For example, if `total` and `count` are integers, but we want a floating point result when dividing them, we can cast `total`:

```
int total = 10;
result = (double) total / count;
```

Which is NOT the same as:

```
result = (double) (total / count);
```

Positive floating-point numbers can be rounded to the nearest integer as `(int)(x + 0.5)`, negative numbers as `(int)(x - 0.5)`.

```
double x = 14.6;
int y = (int) (x + 0.5);
```

```
double x = -9.7;
int y = (int) (x - 0.5);
```

- Given the declarations below, find the result of each expression. Create a Java program to check your answers.

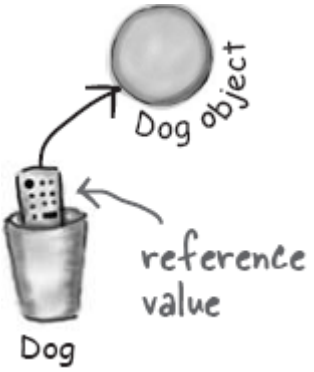
```
int a = 3, b = 10, c = 7;
double w = 12.9, y = 3.2;

a. a + b * c

b. a - b - c

c. a / b

d. b / a

e. a - b / c

f. w / y

g. y / w

h. a + w / b

i. a % b / y

j. b % a

k. w %  y
```

2.4 Object Creation

A variable holds either a `primitive` type or a *reference* to an object. A *class name* can be used as a type to declare an *object reference variable*.

```
Dog myDog;        // Tells the JVM to allocate space for a reference
                  // variable and names that variable myDog. At this
                  // point, myDog contains null.
```

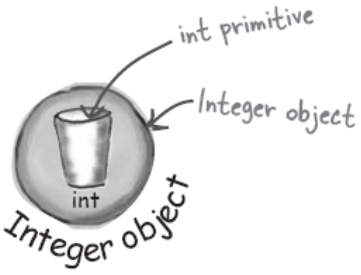
No object is created with this declaration. Generally, we use the `new` operator to create an object.




```
myDog = new Dog();     // Tells the JVM to allocate space
                       // for a new Dog object and store the
                       // location of the object in the
                       // reference variable myDog.
```

Creating an object is called *instantiation.* An object is an *instance* of a particular class.

Wrapper Classes



A *wrapper class* represents a particular primitive type.

For example

```
Integer ageObj = new Integer (20);
```

uses the Integer class to create an object which effectively represents the integer 20 as an object. This is useful when a program requires an object instead of a primitive type.

`Integer.MIN_VALUE`	is a constant value holding the minimum value of an int -2^{31}.
`Integer.MAX_VALUE`	is a constant value holding the maximum value of an int $2^{31}-1$.

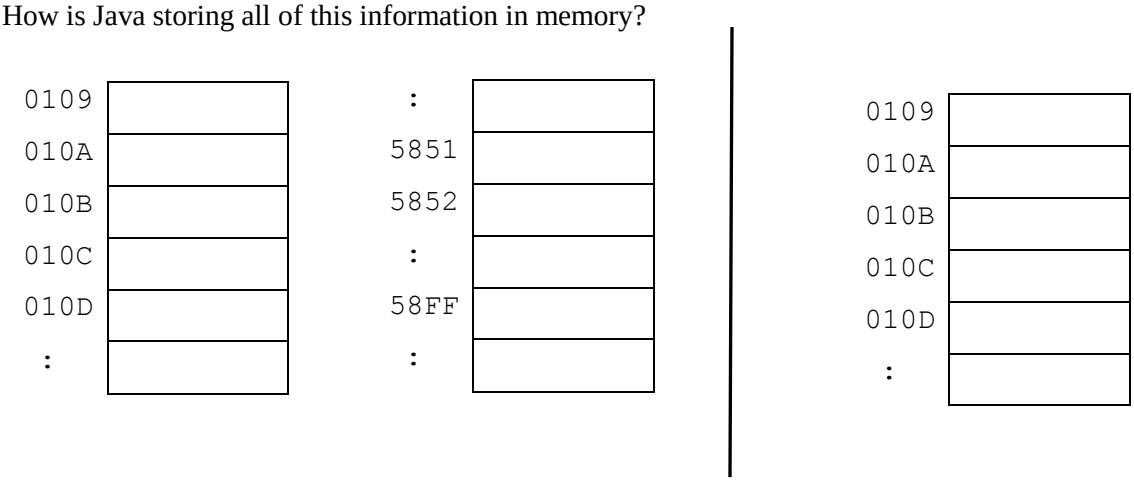
You will need to use these constants on the AP exam when developing an algorithm to sort a list of numbers.

2.5 String class

Strings are also objects. Because strings are so common, we don't have to use the `new` operator to create a String object.

```
String title;

title = new String ("Java Software Solutions");

title = "Oak Ridge High School";
```

How is Java storing all of this information in memory?



This is special syntax that works only for strings. Once an object has been instantiated, we can use the *dot operator* . to invoke its methods or instance variables that are marked public.

```
title.length()
```

Once a String object is created, its value cannot be lengthened or shortened, nor can any of its characters change. Thus we say that a `String` object is *immutable.* The `String` class has several methods. For the AP exam, you need to know the methods listed in the APCS Quick Reference guide.

```
String apcs = "APCS is the best class ever!";

System.out.println (apcs.length());
String apcs1 = apcs.substring(0,3);

System.out.println(apcs1);
String apcs2 = apcs.substring(12);
System.out.println(apcs2);
```

Output:

Some other useful String methods are `toLowerCase( )`, which changes all the characters of a string to lowercase. `toUpperCase( )` changes all of the characters of a string to uppercase.

```
String city = "El Dorado Hills";
city = city.toUpperCase( );
System.out.println (city);
```

Output:

2.6 Math Class

Static Methods

Some methods can be invoked through the *class name*, instead of through an object of the class. These methods are called *class methods* or *static methods.* Java is object-oriented, but once in a while you have a special case, typically a utility method (like the `Math` methods), where there is no need to have an instance of the class. The keyword `static` lets a method run ***without any instance of the class***. A static method means "behavior not dependent on an instance variable, so no instance/object is required. Just the class." The `Math` class contains many static methods, providing various mathematical functions, such as absolute value, trigonometry functions, square root, etc.

```
temp = Math.cos(90) + Math.sqrt(delta);
```

The `cos` and `sqrt` methods are called directly from the `Math` class. No object of `Math` type was created in order to use the methods.

For the AP exam, static methods are always invoked through a *class* never an *object* (i.e., `ClassName.staticMethod ( )`, not `obj.staticMethod( )`).

Math Class

The `Math` methods required for the AP exam are listed below and in the APCS quick reference guide.

```
static int abs(int x)
static double sqrt(double x)
static double abs (double x)
static double pow(double base, double exponent)
static double random()     // returns a double 0 ≤ x < 1
```

- Given four `int` values, `x1, x2, y1, y2`, write the code to compute the distance between the two points (`x1, y1`) and (`x2, y2`), storing the result in the double distance.

Random Numbers

Use the `random()` method of the `Math` class, not the `Random` class in the `java.util` package. `Math.random()` returns a random number between `0` and `.99`.

```
int randomNum = (int) (Math.random() * 5);
```

- Write the code to generate a random integer between 1 and 10:

- Write the code to generate a random multiple of 45 for degrees in a circle. (0, 45, 90, 135, 180, 225, 270, 315):

- Write the code to generate a random real number greater than or equal to 4.0 and less than 6.0:

2.7 Scanner Class

The Scanner class is used to get input from the user, allowing a program to be interactive. It is part of the `java.util` package. In order to use this class, you must import the package at the top of your source code, (`import java.util.*` or `import java.util.Scanner`). First a Scanner object is created:

```
Scanner scan = new Scanner (System.in);
```

Then various methods can be used to read different types of data from the keyboard:

```
int num = scan.nextInt();

String guess = scan.nextLine();          [ or scan.next(); ]

double amount = scan.nextDouble();
```

- Write a class `Birthday` to prompt the user for their name and age, putting them into a `String` and `int` variable. Print out a message including their name and the year they were born.

- Complete the following program to read three integers and print the average.

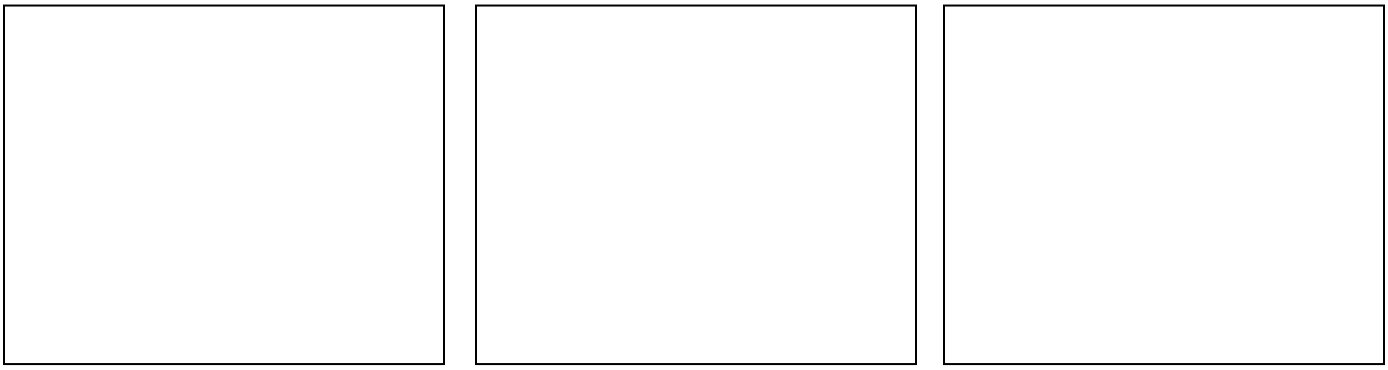
```
import java.util.Scanner;
public class Average
{
  public static void main(String[] args)
  {
    int val1, val2, val3;
    double average;
    Scanner scan = new Scanner(System.in) ;

    System.out.println("Please enter three integers and " +
                       "I will compute their average");









   }  // end of main method
} // end of class
```

- Carefully study the following program and find and correct all of the syntax errors.

```
// File:     Errors.java
// Purpose:  A program with lots of syntax errors
//           Correct all of the errors (STUDY the program carefully!!)

#import java.util.Scanner;

public class errors
{
   public static void main (String[] args)
   {
       String Name;  / Name of the user
       int number;
       int numSq;
       Scanner scan = new Scanner(System.in);

       System.out.print ("Enter your name, please: ")
       Name = scan.nextInt();

       System.out.print ("What is your favorite number?);
       number = scan.nextInt();

       numSq = number * number;

       System.out.println (Name  ", the square of your number is "
                           numSquared);

}
```

Classes and Objects – In Summary

- Objects have *state* (what it knows) called *instance variables* and *behavior* (what it does) called *methods*.
- Classes are text files containing the code for the object. Class names and the file name must be exactly the same (case sensitive). There are hundreds of classes available for your use. A few of the ones you will use often are:



```
import java.util.Scanner;

public class MyProgram
{
      public static void main (String[] args)
      {
            Scanner scan = new Scanner (System.in);
            int base = 0,exp = 0;
            System.out.println ("Enter a base number: ");
            base = scan.nextInt();
            System.out.println ("Enter an exponent number: ");
            exp = scan.nextInt();
            double answer = Math.pow (base, exp);
            System.out.println ("The answer is: " + answer);
      }
}
```

Java Applets & Graphics

A Java application is a stand-alone program with a main method (like the ones we've seen so far).

A Java *applet* is a program that is intended to transported over the Web and executed using a web browser. The commands to execute the applet are written in an html file using Notepad.

An applet also can be executed using the *appletviewer* tool of the Java Software Development Kit. An applet doesn't have a `main` method.

You can run your applets from BlueJ. Make sure your `.java`, `.class` and `.html` files are in the same folder and all have the same name (example: `Shapes.java`, `Shapes.class`, `Shapes.html`).

Graphics class

Shapes can be drawn using methods of the `Graphics` class in the `java.awt` package. A shape can be filled or unfilled, depending on which method is invoked. The method parameters specify coordinates and sizes. Recall from Chapter 1 that the Java coordinate system has the origin in the top left corner. Shapes with curves, like an oval, are usually drawn by specifying the shape's *bounding rectangle.* An arc can be thought of as a section of an oval.

```
drawLine (int x1, int y1, int x2, int y2)   // Draws a line from point (x1,y1) to (x2,y2)

drawRect (int x, int y, int width, int height)    // Draws a rectangle with upper left corner
                                                  // (x, y), and dimensions width and height.

drawOval (int x, int y, int width, int height)    // Draws an oval in the rectangle with
                                                  // upper left corner (x, y), and dimensions
                                                  // width and height.
```

```
page.drawLine(10, 20, 150, 45);
```

```
page.drawRect (50, 20, 100, 40);
```

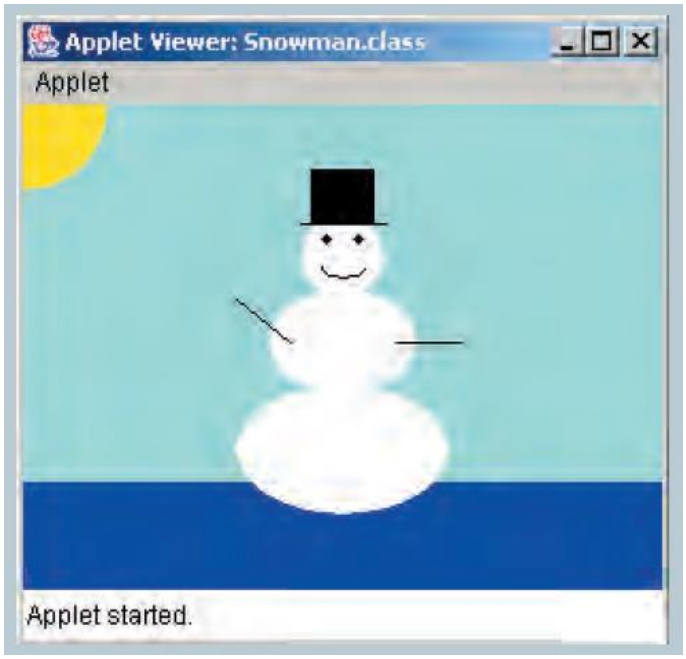
```
page.drawOval (175, 20, 50, 80);
```

Color class

A color is defined in a Java program using an object created from the Color class. The Color class also contains several static predefined colors, including:

Object	RGB Value
Color.black	0, 0, 0
Color.blue	0, 0, 255
Color.cyan	0, 255, 255
Color.orange	255, 200, 0
Color.white	255, 255, 255
Color.yellow	255, 255, 0

Look at the `Color` class in the Java API for the different color constants.




```
// ********************************
// Snowman.java
//
// ********************************
import java.applet.Applet;
import java.awt.*;
public class Snowman extends Applet
{
//----------------------------------------
// Draws a snowman.
//----------------------------------------
     public void paint (Graphics page)
     {
         final int MID = 150;
         final int TOP = 50;
         setBackground (Color.cyan);
         page.setColor (Color.blue);
         page.fillRect (0, 175, 300, 50); // ground
         page.setColor (Color.yellow);
         page.fillOval (-40, -40, 80, 80); // sun
         page.setColor (Color.white);
         page.fillOval (MID-20, TOP, 40, 40); // head
         page.fillOval (MID-35, TOP+35, 70, 50); // upper torso
         page.fillOval (MID-50, TOP+80, 100, 60); // lower torso
         page.setColor (Color.black);
         page.fillOval (MID-10, TOP+10, 5, 5); // left eye
         page.fillOval (MID+5, TOP+10, 5, 5); // right eye
         page.drawArc (MID-10, TOP+20, 20, 10, 190, 160); // smile



         page.drawLine (MID-25, TOP+60, MID-50, TOP+40); // left arm
         page.drawLine (MID+25, TOP+60, MID+55, TOP+60); // right arm
         page.drawLine (MID-20, TOP+5, MID+20, TOP+5); // brim of hat
         page.fillRect (MID-15, TOP-20, 30, 25); // top of hat
      }
}
```

Snowman.html

```
<html>
<applet code="Snowman.class" width=600 height=400>
</applet>
</html>
```