Chapter 3 Notes (Objects, methods, classes, DecimalFormat)

- A class API consists of the class method names, their return values, and their argument lists.

- The program that uses a class is called the client of the class.

- The identifier of the object is called the object reference. Creating an object of a class is called an instance of the class. Many objects can be instantiated from one class.

(object reference : class :: variable : primitive data type)

- The data associated with an object of a class are called instance variables or fields and can be variables or constants of any primitive data type (byte, short, int, long, float, double, char, and Boolean), or they can be objects of a class.

- Methods are operations for a class such as: set the values of the data, retrieve the current values of the data, etc. Invoking a method on an object is referred to as, “calling the method”. With a few exceptions, only the methods of a class can directly access or change the instance variables of an object.

- Together, the fields and methods of a class are called its members.

- In essence, a class is a new data type, which is created by combining items of Java primitive data types and objects of classes. Just as primitive data types can be manipulated using the arithmetic operators (+, - , \*, /, and %), an object can be manipulated by calling the methods of its class.

- The data of a class is typically declared to be private. In this case, only the methods of the class can change the data of an object, and thus the methods provide a protective layer around the data. The methods of the class provide the only interface to set or change the data values from outside the class.

- We don’t need to know the implementation details of a class in order to use it in our program. This is one of the best parts of object-oriented programming: all we need to know is the class Application Programming Interface (API), that is, how to instantiate objects and how to call the methods of the class.

- A class describes a template for instantiating objects. An object must be instantiated before it can be used. In order to instantiate an object and how to call methods of a class we must know the API of a class.

- Instantiating an object consists of defining an object reference – which will hold the memory address of the object in memory – and calling a special method of the class called a constructor, which has the same name of the class. The job of the constructor is to assign initial values to the data of the class.

public class Constructors

{

public static void main( String[] args);

{

SimpleDate independenceDay;

independenceDay = new SimpleDate( 7, 4, 1776);

SimpleDate nextCentury = new SimpleDate( 1, 1, 2101);

SimpleDate defaultDate = new SimpleDate( );

}

}

- Once an object reference has been declared, we instantiate the object using the following syntax:

objectReference = new ClassName ( argument list );

- This calls a constructor of the class to initialize the data. The argument list consists of a comma-separated list of initial data values to assign to the object. Classes often provide multiple constructors with different argument lists. Depending on which constructor we call, our argument list – that is, the number of arguments and their data types – must match one of the constructors’ argument lists.

- Once an object is instantiated, we can use the object by calling its methods. Authors publish their API so that their clients know what methods are available and how to call those methods.

- A method may or may not return a value, as indicated by a data type, class type, or the keyword void in front of the method name. If the method returns a value, then the data type or class type of its return value will precede the method’s name. The call to a value-returning method can be used in an expression. When the method finishes executing, its return value will replace the method call in the expression.

- If the keyword void precedes the method name, the method does not return a value. Because methods with a void return type have no value, they cannot be used in an expression; instead, a method call to a method with a void return type is a complete statement.

- Public is a keyword we will see preceding the method call in an API. This keyword means that any client of the class can call this method. If the keyword private precedes the method name, only other methods of that class can call that method.

- To call a method for an object of a class, we use dot notation, as follows:

objectReference.methodName ( arg1, arg2, arg3 . . .);

- The object reference is followed immediately by a dot, which is followed immediately by the method name.

- An object reference points to the data of an object. The object reference and the object data are distinct entities. An object can have more than one object reference pointing to it, or an object can have no object references pointing to it.

Text

Description automatically generated

Text, letter

Description automatically generated

- When an object reference is first declared, but has not yet been assigned to an object, its value is a special literal value, null. If we attempt to call a method using an object reference whose value is null, Java generates either a compiler error or a run-time error called an exception.

- One way to indicate to the garbage collector that our program is finished with an object is to set its object reference to null. Obviously, once an object method has the value null, it can no longer be used to call methods.

- Java provides more than 2,000 predefined classes that we can use to add functionality to our program. In this chapter, we’ll discuss a few commonly used Java classes:

String , which provides a data type for character sequences along with methods for searching and manipulating strings.

Random , which generates random numbers.

Scanner , which provides methods for reading input from the keyboard.

System and PrintStream , which provide data members and methods for printing data on the Java console.

DecimalFormat and NumberFormat , which allows us to format numbers for output.

Math , which provides methods for performing mathematical operations.

Object wrappers, which provide an object equivalent to primitive data types so they can be used in our programs as if they were objects.

- The Java classes are arranged in packages and grouped according to functionality.

- To use a class from the Java Class Library that is not in the java.lang package, we need to tell the compiler in which package the class resides; in other words, we need to tell the compiler where to find the class. To do this, we include an import statement in our program. The import statement is inserted at the top of the program after our introductory comments, but before the class statement that begins the program.

- For example, if we want to use the DecimalFormat class to format a floating-point number for output, we would import the DecimalFormat class from the java.text package as follows:

Import java.text.DecimalFormat;

- If we’re using more than one class from a package, we can import all those classes we use by replacing the class name with an asterisk:

Import java.text.\*;

Text, letter

Description automatically generated

Text

Description automatically generated

- Demonstration of String methods:

In this program the first constructor:

String( String str )

Allocates a String object and sets it’s value to the sequence in the argument str, which can be a String object or a String literal. Line 8 instantiates the String s1 and sets its value to “OOP in Java”.

Text

Description automatically generated

The second constructor

String( )

Creates an empty String. - in other words, a String containing no characters. We can add characters to the String later. This constructor will come in handy for programs where we build up our output, piece by piece. Line 19 uses the second constructor to instantiate an empty String named empty.

- Note that the String concatenation operator is the same character as the addition arithmetic operator. In some cases, we need to make clear to the compiler which operator we want to use. For example, this statement uses both the String concatenation operator and the addition operator:

System.out.println(“The sum of 1 and 2 is” + (1 + 2) );

Notice that we put 1 + 2 inside the parentheses to let the compiler know that we want to add two ints using the addition arithmetic operator. The addition will be performed first because of the higher operator precedence of parentheses. Then it will become clear to the compiler that the other operator is intended to be a String concatenation operator because its operands are a String and an int.

(Check pages 109-116 for String methods)

- The DecimalFormat class allows us to specify the number of digits to display after the decimal output and to add dollar signs, commas, and percentage signs (%) to our output. The DecimalFormat class is part of the java.text package, so to use the DecimalFormat class, we include the following import statement in our program:

import java.text.DecimalFormat;

- We can instantiate a DecimalFormat object using a simple constructor that takes a String object as an argument. This String object represents how we want our formatted number to look when it’s printed. The pattern that we use to instantiate the DecimalFormat object consists of special characters and symbols to create a “picture” of how we want the number to look when printed.

Timeline

Description automatically generated with low confidence

Text, letter

Description automatically generated

Text, letter

Description automatically generatedText

Description automatically generatedA picture containing text, whiteboard

Description automatically generated