Defining Classes and Methods

Chapter 5
Objectives

• Describe concepts of class, class object
• Create class objects
• Define a Java class, its methods
• Describe use of parameters in a method
• Use modifiers public, private
• Define accessor, mutator class methods
• Describe information hiding, encapsulation
• Write method pre- and postconditions
Objectives

• Describe purpose of javadoc
• Draw simple UML diagrams
• Describe references, variables, parameters of a class type
• Define boolean-valued methods such as equals
Class and Method Definitions: Outline

• Class Files and Separate Compilation
• Instance Variables
• Methods
• The Keyword this
• Local Variables
• Blocks
• Parameters of a Primitive Type
Class and Method Definitions

• Java program consists of objects
  ▪ Objects of class types
  ▪ Objects that interact with one another

• Program objects can represent
  ▪ Objects in real world
  ▪ Abstractions
Class and Method Definitions

• Figure 5.1  A class as a blueprint

Class Name: Automobile

Data:
    amount of fuel
    speed
    license plate

Methods (actions):
    accelerate:
        How: Press on gas pedal.
    decelerate:
        How: Press on brake pedal.
Class and Method Definitions

• Figure 5.1 ctd.

Objects that are instantiations of the class **Automobile**
Class and Method Definitions

• Figure 5.2 A class outline as a UML class diagram

```
Automobile

- fuel: double
- speed: double
- license: String

+ accelerate(double pedalPressure): void
+ decelerate(double pedalPressure): void
```
Class Files and Separate Compilation

• Each **Java** class definition usually in a file by itself
  ▪ File begins with name of the class
  ▪ Ends with `.java`

• Class can be compiled separately

• Helpful to keep all class files used by a program in the same directory
Dog class and Instance Variables

- View sample program, listing 5.1
  ```java
class Dog
```
- Note class has
  - Three pieces of data (instance variables)
  - Two behaviors
- Each instance of this type has its own copies of the data items
- Use of `public`
  - No restrictions on how variables used
  - Later will replace with `private`
Using a Class and Its Methods

• View sample program, listing 5.2
  ```java
class DogDemo

Name: Balto
Breed: Siberian Husky
Age in calendar years: 8
Age in human years: 52

Scooby is a Great Dane.
He is 42 years old, or 222 in human years.
```
Sample screen output
Methods

• When you use a method you "invoke" or "call" it

• Two kinds of Java methods
  ▪ Return a single item
  ▪ Perform some other action – a void method

• The method main is a void method
  ▪ Invoked by the system
  ▪ Not by the application program
Methods

• Calling a method that returns a quantity
  ▪ Use anywhere a value can be used

• Calling a void method
  ▪ Write the invocation followed by a semicolon
  ▪ Resulting statement performs the action defined by the method
Defining **void** Methods

- Consider method **writeOutput** from Listing 5.1

```java
public void writeOutput()
{
    System.out.println("Name: " + name);
    System.out.println("Breed: " + breed);
    System.out.println("Age in calendar years: " + age);
    System.out.println("Age in human years: " +
                     getAgeInHumanYears());
    System.out.println();
}
```

- Method definitions appear inside class definition
  - Can be used only with objects of that class
Defining void Methods

• Most method definitions we will see as **public**
• Method does not return a value
  ▪ Specified as a **void** method
• Heading includes parameters
• Body enclosed in braces `{   }`
• Think of method as defining an action to be taken
Methods That Return a Value

• Consider method `getAgeInHumanYears()`

```java
public int getAgeInHumanYears()
{
    int humanAge = 0;
    if (age <= 2)
    {
        humanAge = age * 11;
    }
    else
    {
        humanAge = 22 + ((age-2) * 5);
    }
    return humanAge;
}
```

• Heading declares type of value to be returned

• Last statement executed is `return`
Second Example – Species Class

• Class designed to hold records of endangered species

• View the class listing 5.3

```java
class SpeciesFirstTry
```
- Three instance variables, three methods
- Will expand this class in the rest of the chapter

• View demo class listing 5.4

```java
class SpeciesFirstTryDemo
```
The Keyword **this**

- Referring to instance variables outside the class — must use
  - Name of an object of the class
  - Followed by a dot
  - Name of instance variable
- Inside the class,
  - Use name of variable alone
  - The object (unnamed) is understood to be there
The Keyword **this**

- Inside the class the unnamed object can be referred to with the name **this**
- Example
  ```java
  this.name = keyboard.nextLine();
  ```
- The keyword **this** stands for the receiving object
- We will see some situations later that require the **this**
Local Variables

- Variables declared inside a method are called *local* variables
  - May be used only inside the method
  - All variables declared in method `main` are local to `main`

- Local variables having the same name and declared in different methods are different variables
Local Variables

• View sample file, listing 5.5A
  class BankAccount

• View sample file, listing 5.5B
  class LocalVariablesDemoProgram

• Note two different variables newAmount
  ▪ Note different values output

With interest added, the new amount is $105.0
I wish my new amount were $800.0
Blocks

• Recall compound statements
  ▪ Enclosed in braces `{ }`

• When you declare a variable within a compound statement
  ▪ The compound statement is called a block
  ▪ The scope of the variable is from its declaration to the end of the block

• Variable declared outside the block usable both outside and inside the block
Parameters of Primitive Type

• Recall method declaration in listing 5.3
  ▪ Note it only works for 10 years
  ▪ We can make it more versatile by giving the method a parameter to specify how many years

• Note sample program, listing 5.6

```java
public int getPopulationIn10()
{
    int result = 0;
    double populationAmount = population;
    int count = 10;
}
```

class SpeciesSecondTry
Parameters of Primitive Type

• Note the declaration
  
  ```java
  public int predictPopulation(int years)
  ```
  ▪ The formal parameter is `years`

• Calling the method
  
  ```java
  int futurePopulation =
      speciesOfTheMonth.predictPopulation(10);
  ```
  ▪ The actual parameter is the integer 10

• View sample program, listing 5.7

  ```java
  class SpeciesSecondClassDemo
  ```
Parameters of Primitive Type

• Parameter names are local to the method
• When method invoked
  ▪ Each parameter initialized to value in corresponding actual parameter
  ▪ Primitive actual parameter cannot be altered by invocation of the method
• Automatic type conversion performed

\[
\text{byte} \rightarrow \text{short} \rightarrow \text{int} \rightarrow \text{long} \rightarrow \text{float} \rightarrow \text{double}
\]