# Classes

## Introduction

This unit covers how to design and build your own custom classes. A **class** is a definition file for an Object. It defines what type of data an Object will store and what methods it will have.

## Classes Overview

### Files

A Class is typically written in its own file. When making a new file for a class, the file name must match the name of the class being built. For example, a class named Person would need to be in a file called Person.

### Parts

A class consists of the following three parts: attributes, constructors and methods. These parts can appear in any order, but they are generally written in the order attributes, constructors, and then methods.

### Attributes

**Attributes** store data for your object. For instance, a person would need attributes like first name, last name, social security number, etc.

### Constructors

**Constructors** are special methods used for building an object. Constructors are responsible for assigning the initial values of the attributes.

### Methods

Classes contain many methods for performing actions on its data.

## Making Attributes

A class is not required to have attributes, but it rare for them not to have any. When creating an attribute you have to decide what **access level** and what **modifiers** you want to put on the data. Access levels determine how much access there is to an attribute or method outside of the file. Modifiers set properties for attributes or methods.

### Access Levels

- public – the attribute can be modified and viewed outside of the class's file.
- private – the attribute cannot be modified or viewed outside of the class's file.
- protected – the attribute can only be modified by subclasses or classes in the same package

**Modifiers**

- final – the value of the attribute cannot be changed after it has been initialized.
- static – Sets the attribute to be a **class attribute**. Class attributes are created once and are shared between all **instances** of a class. When an Object is created from a class that Object is considered an instance of the class. Non-static attributes are called **instance attributes**. Instance attributes are created each time an instance of the class is created. Each instance of a class will have its own copy of each instance attribute.

**Format for creating an Attribute:**

accessLevel modifiers type attributeName;

**Examples:**

```
public static final double PI = 3.14;
private String name;
private int age = 0;
```

# Making Constructors

**Rules for creating constructors:**

- Constructors are public
- Constructors do not have a return type; not even void
- Constructor can receive parameters
- Constructors can be overloaded
- A constructor that receives no parameters is called a **default constructor**. If you do not write any constructors the class will have a hidden default constructor that sets all primitives to 0 and all Objects to null.

**The format for a constructor is as follows:**

```
public className(parameters)
{
    code
}
```

# Methods

**Accessors**

**Accessors** are methods that allow public viewing access to private attributes.

**Format for an accessor:**

```
public returnType getVariableName()
{ return variableName; }
```

**Example:**

```
public int getAge()
{ return age; }
```

### Mutators

**Mutators** are methods that allow private attributes to be changed publicly.

**Format for a mutator:**

```
public void setVariableName(variableType variableName)
{ this.variableName = variableName; }
```

**Example:**

```
public void setAge(int age)
{ this.age = age; }
```

### Other Methods

Classes usually methods other than just accessors and mutators. For example, Scanner has many methods for retrieving data from the keyboard and String has methods for searching its contents. When writing a class you can create any method you want.

**Format for methods:**

```
accessLevel modifiers returnType methodName(parameters)
{
        code
}
```

**Access Levels:**

- public – the method can be used outside of the class's file.
- private – the method cannot be used outside of the class's file.
- protected – the method can be used by subclasses or classes with the same package

**Modifiers:**

- final – It is different than the final of an attribute and it will be covered in the next chapter.
- static – The makes the method a **class method**. A class method can be accessed through the classes name or through an instance. A class method can only use static attributes. By default all methods are **instance methods**. Instance methods are methods that can only be accessed through an instance of the class.

## Working with Instances

After a class has been defined new objects can be created from it..

**Format for creating an instance of a class:**

```
className  variableName = new className(parameters);
```

**Accessing public attributes:**

```
variableName.attributeName
```

**Changing public attributes**

variableName.attributeName = newValue;

**Accessing private attributes:**

variableName.accessor();

**Changing private attributes:**

variableName.mutator( newValue);

**Accessing public class methods:**

variableName.methodName(parameters);

## Working with Class Attributes & Methods

Class attributes and methods can be accessed with instances or without them.

**Accessing public class attributes:**

className.attributeName

**Changing public class attributes:**

className.attributeName = newValue;

**Accessing public class methods:**

className.methodName(parameters);

## this

**this** is a keyword used to access the current instance of a class from within a class file.

**Using this to access an attribute:**

this.attributeName

**Using this to call another constructor from a constructor:**

this(parameter);

# Package

A **Package** is a group of related classes. These classes have unrestricted access to each other's protected methods and attributes.

**Benefits of Packages:**

- Other programmers can see that the classes are related
- It helps further organize your code
- The names of the classes will not conflict with classes in other packages
- Having unrestricted access to each other's protected methods and attributes

**Names Packages:**

- Names should be in all lowercase
- Do not begin with java or javax
- Use periods for spaces

**Example Names:**

- game.graphics
- delivery.drivers

**Setting the Package of a File**

On the first line of you file, above the imports write package followed by the name of the package and a semicolon.

**Example:**

```
package game.graphics;

import java.awt.graphics;
import java.awt.Point;
…
```

# Examples

Shape Perimeter

# Terms

| | |
|---|---|
| **Accessor** | A public method that allows access to a private attribute. |
| **Access Level** | Determines if a method or attribute can be accessed outside of a class file. |
| **Attribute** | A piece of data for a class. |
| **Class** | A file that defines an new Object type. |
| **Class Attribute** | A attribute that is shared between all instances of a class and can be accessed without an instance of the class. |
| **Class Method** | A method that can be accessed without a instance of the class and can only access class attributes. |
| **Constructor** | A method used for setting the attributes of an object when it is created. |
| **Default Constructor** | A constructor that receives no parameters. A default constructor is generated automatically when no constructors are written. |
| **Instance** | An object created from a class. |
| **Instance Attribute** | An attribute that only exists when there is an instance of a class. Each instance of a class has its own copy of each instance variable. |
| **Instance Method** | A method that can only be accessed through an instance of a class. |
| **Modifiers** | Keywords used to set whether a method/attribute can be modified or if it should class/instance. |
| **Mutator** | A method used to change the value of a private attribute. |
| **Package** | A grouping of related classes. |
| **Private** | Cannot be accessed outside of the class file. |
| **Protected** | Can only be accessed by subclasses or classes with the same package. |
| **Public** | Can be accessed outside of the class file. |
| **Static** | Sets whether a method/attribute is instance or class. |