**Administrative**

**Today’s session**

Homework 3 key

Classes and objects

Instance and local variables

Access modifiers

**Session Topics**

**Homework 3 key**

● The Homework 3 key is available on Blackboard.

**Classes and objects**

● A **class** is a blueprint.

● A class consists of one or more **members** including:

✓ Fields – the data stored in an object.

✓ Methods – the actions that run on or within an object.

✓ Nested classes

● Fields are typically invisible to the outside world while methods are typically visible.

● An **object** results from the **instantiation** of a class.

● An object is a *running* version/instance of a class.

● During program execution, an **object**:

✓ Is created from a class.

✓ Its fields and methods are used.

✓ May be explicitly deleted.

● A **Java program** is a set of one or more classes.

● A **running Java program** is a set of one or more objects.

● The computer needs to know where to start a Java program, so one of its classes must include a method called **main**.

● There is typically one class per .java file, and the name of the class matches the name of the file.

● Class example that would be stored in file **MyClass.java**:

public class MyClass

{

// Declare instance variables

private int ivVar1;

private int ivVar2;

// Declare method1

public void method1 ()

{

// Declare local variables

int lvVar1;

int lvVar2;

…do something here …

}

// Declare method2

public void method2 ()

{

// Declare local variables

int lvVar1;

int lvVar2;

…do something here …

}

}

● See **Classes – one file** sample application on Blackboard.

● See **Classes – two files** sample application on Blackboard.

**Instance and local variables**

● There are two types of variables within an object.

● Together they hold the data for an individual object.

● An **instance variable** is declared outside any methods so it is available throughout an object.

● A **local variable** is declared inside a method and so is only available within that method.

● Local and instance variables example:

public class CounterClass

{

// Declare instance variables

private int ivCount1;

// this variable can be used throughout the class

private int ivCount2;

// this variable can be used throughout the class

// showCounts method

public void showCounts()

{

System.out.println("ivCount1 is " + ivCount1);

System.out.println("ivCount2 is " + ivCount2);

}

// addCount1 method

public void addCount1()

{

int lvCount = 30;

// this variable can only be used in this method

ivCount1 = ivCount1 + 1;

System.out.println("lvCount is " + lvCount);

}

// addCount2 method

public void addCount2()

{

int lvCount = 40;

// this variable can only be used in this method

ivCount2 = ivCount2 + 1;

System.out.println("lvCount is " + lvCount);

}

}

● See **Classes – one file** sample application on Blackboard.

● See **Classes – two files** sample application on Blackboard.

**Access modifiers**

● An **access modifier** determines how a class or class member may be accessed outside the class.

● There are four access modifiers that may be used:

|  |  |
| --- | --- |
| Modifier | Purpose |
| public | To make the class or class member visible everywhere. |
| protected | To make the class member visible within the package and any child class outside the package. |
| package-private (no modifier) | To make the class or class member only visible within the package. |
| private | To make the class member only visible within the class. |

● A class may only be declared with an access modifier of **public** or **package-private**.

● A class member (field or method) may be declared with any access modifier.

● This table shows the visibility of class members:

| Modifier | 1 | 2 | 3 | 4 |
| --- | --- | --- | --- | --- |
| public | Yes | Yes | Yes | Yes |
| protected | Yes | Yes | Yes | No |
| (none) | Yes | Yes | No | No |
| private | Yes | No | No | No |

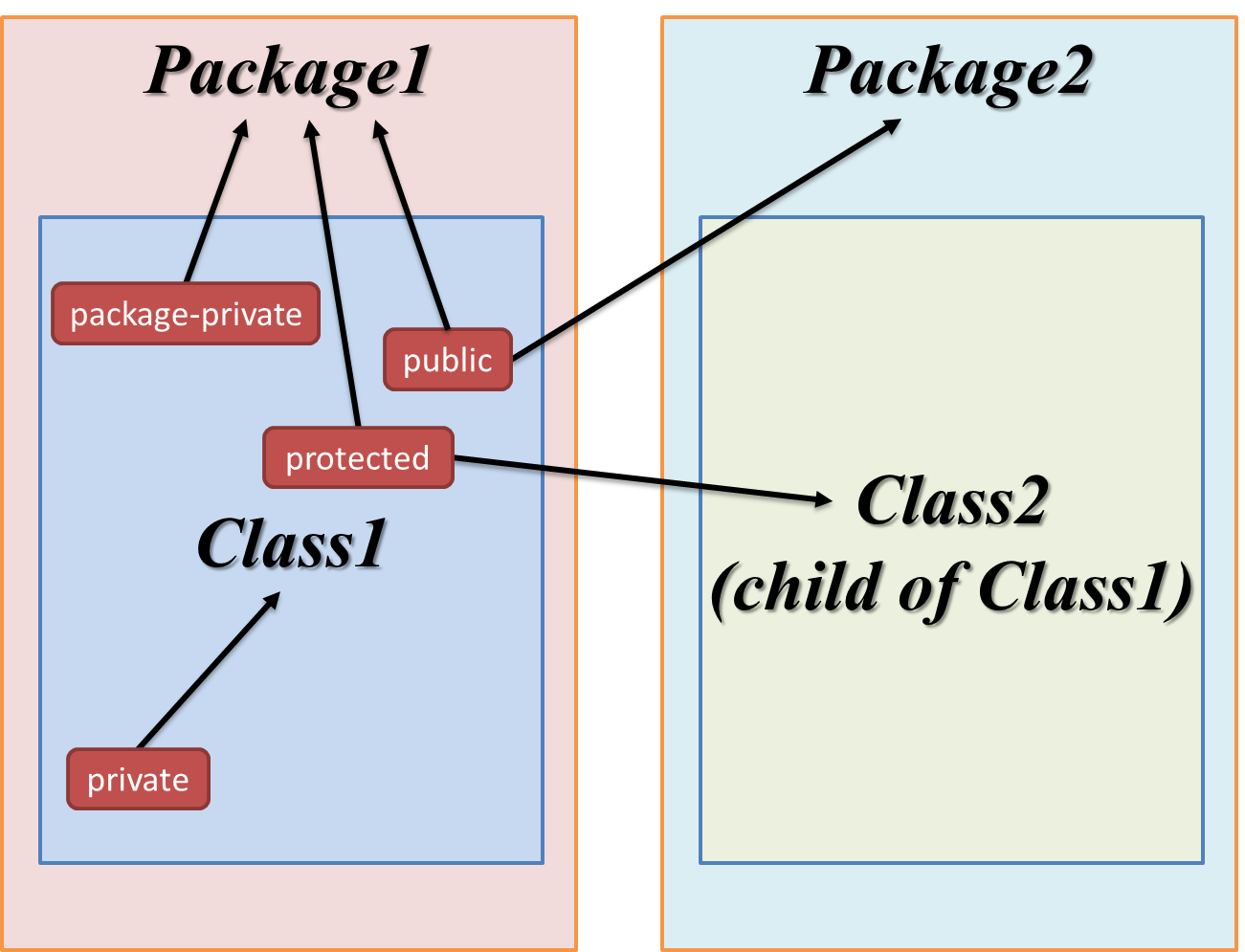
1) Is the class member visible in the class?

2) Is the class member visible in the package?

3) Is the class member visible in subclasses of the class declared outside the package?

4) Is the class member visible everywhere?

● Here is an access modifier graphic:



● To reference a field or method outside its class, an object is required:

<object-name>.<class-variable>

<object -name>.<class-method>

● Generally, fields are declared private while methods are declared public.

● A **class** with modifier has syntax:

// Define class "public"

public class <class-name>

{

…

}

OR

// Define class "package-private"

class <class-name>

{

…

}

● A **field** with modifier has syntax (these all declare instance variables):

// Define field "public"

public <data-type> <variable-name>;

// Define field "protected"

protected <data-type> <variable-name>;

// Define field "package-private"

<data-type> <variable-name>;

// Define field "private"

private <data-type> <variable-name>;

● A **method** with modifier has syntax:

// Define method "public"

public <data-type> <method-name> (<parameter-list>)

{

…

}

OR

// Define method "protected"

protected <data-type> <method-name> (<parameter-list>)

{

…

}

OR

// Define method "package-private"

<data-type> <method-name> (<parameter-list>)

{

…

}

OR

// Define method "private"

private <data-type> <method-name> (<parameter-list>)

{

…

}

● See **Classes – one file** sample application on Blackboard.

● See **Classes – two files** sample application on Blackboard.