Data Scope

- The *scope* of data is the area in a program in which that data can be referenced (used)
- Data declared at the class level can be referenced by all methods in that class
- Data declared within a method can be used only in that method
- Data declared within a method is called *local data*
- In the `Die` class, the variable `result` is declared inside the `toString` method -- it is local to that method and cannot be referenced anywhere else

Example

- **`Demonstrates a constructor calling another method`**
- **`toString method`**

[Date.java]

[TestDate.java]

Quick Check

What is the relationship between a class and an object?

Quick Check

What is the relationship between a class and an object?

A class is the definition/pattern/blueprint of an object. It defines the data that will be managed by an object but doesn't reserve memory space for it. Multiple objects can be created from a class, and each object has its own copy of the instance data.

Quick Check

Where is instance data declared?

What is the scope of instance data?

What is local data?

Quick Check

Where is instance data declared?

At the class level.

What is the scope of instance data?

It can be referenced in any method of the class.

What is local data?

Local data is declared within a method, and is only accessible in that method.

Encapsulation

- We can take one of two views of an object:
  - internal - the details of the variables and methods of the class that defines it
  - external - the services that an object provides and how the object interacts with the rest of the system
- From the external view, an object is an *encapsulated* entity, providing a set of specific services
- These services define the *interface* to the object

Encapsulation

- One object (called the *client*) may use another object for the services it provides
- The client of an object may request its services (call its methods), but it should not have to be aware of how those services are accomplished
- Any changes to the object's state (its variables) should be made by that object's methods
- We should make it difficult, if not impossible, for a client to access an object’s variables directly
- That is, an object should be *self-governing*

Visibility Modifiers

- In Java, we accomplish encapsulation through the appropriate use of *visibility modifiers*
- A *modifier* is a Java reserved word that specifies particular characteristics of a method or data
- Java has three visibility modifiers: `public`, `protected`, and `private`
- The `protected` modifier involves inheritance, which we will not discuss in this course

Visibility Modifiers

- Members of a class that are declared with *public visibility* can be referenced anywhere
- Members of a class that are declared with *private visibility* can be referenced only within that class

Visibility Modifiers

- Public variables violate encapsulation because they allow the client to modify the values directly
- Therefore instance variables should not be declared with public visibility
- It is acceptable to give a constant public visibility, which allows it to be used outside of the class
- Public constants do not violate encapsulation because, although the client can access it, its value cannot be changed

Visibility Modifiers

- Methods that provide the object's services are declared with public visibility so that they can be invoked by clients
- Public methods are also called *service methods*
- A method created simply to assist a service method is called a *support method*
- Since a support method is not intended to be called by a client, it should not be declared with public visibility

Visibility Modifiers

	`public`	`private`
Variables	**Violate encapsulation**	**Enforce encapsulation**
Methods	**Provide services to clients**	**Support other methods in the class**

Quick Check

Why was the `faceValue` variable declared as `private` in the `Die` class?

Why is it ok to declare `MAX` as `public` in the `Die` class?

Quick Check

Why was the `faceValue` variable declared as `private` in the `Die` class?

> By making it private, each `Die` object controls its own data and allows it to be modified only by the well-defined operations it provides.

Why is it ok to declare `MAX` as `public` in the `Die` class?

> `MAX` is a constant. Its value cannot be changed. Therefore, there is no violation of encapsulation.