Salesforce GROW

Module : Object-Oriented Programming in Apex

OurUniversity
EFMD CLIP ACCREDITED

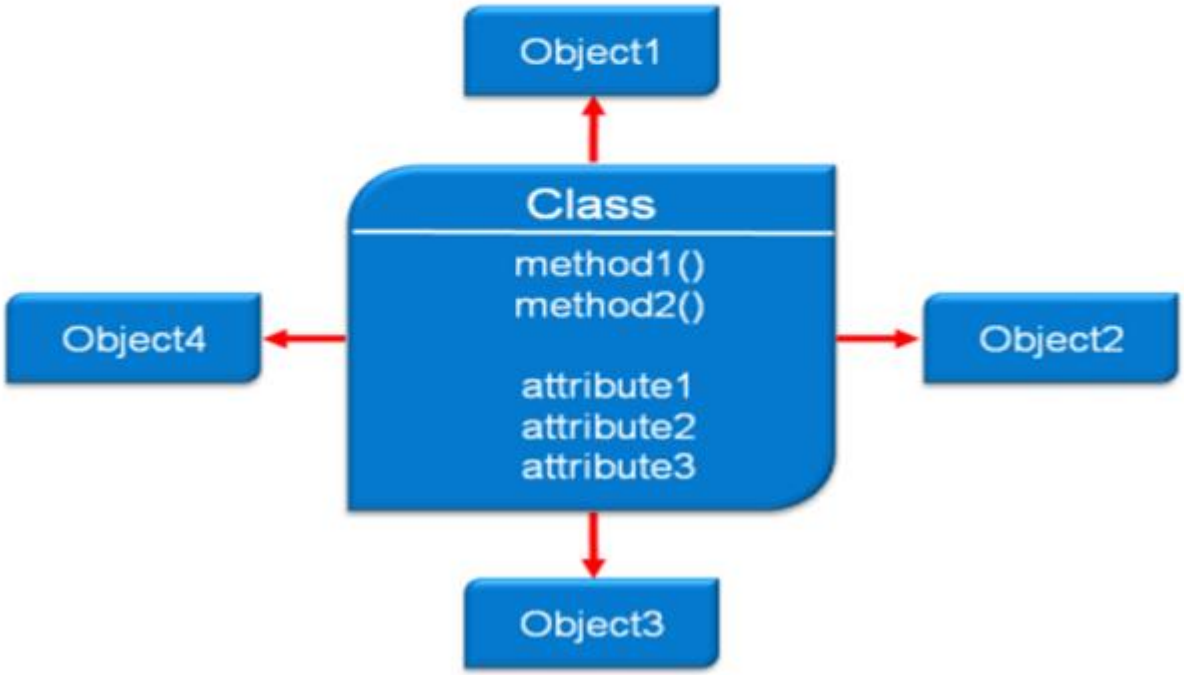
Object-Oriented Programming in Apex

Capgemini

Classes

A class:

- Is a template or blueprint from which objects are created.
- Consists of methods and attributes.
- Is stored with the version of API that is used to compile it.
- May contain other classes, known as inner classes.



Considerations for Creating Classes

An Apex class:

- Should follow the Java naming convention.
- Can be enabled or disabled for profiles.

In Apex:

- Inner classes can only be nested one level deep.
- Static methods and attributes can only be declared in a top-level class definition.
- To create new exception classes, `Exception` class must be extended.

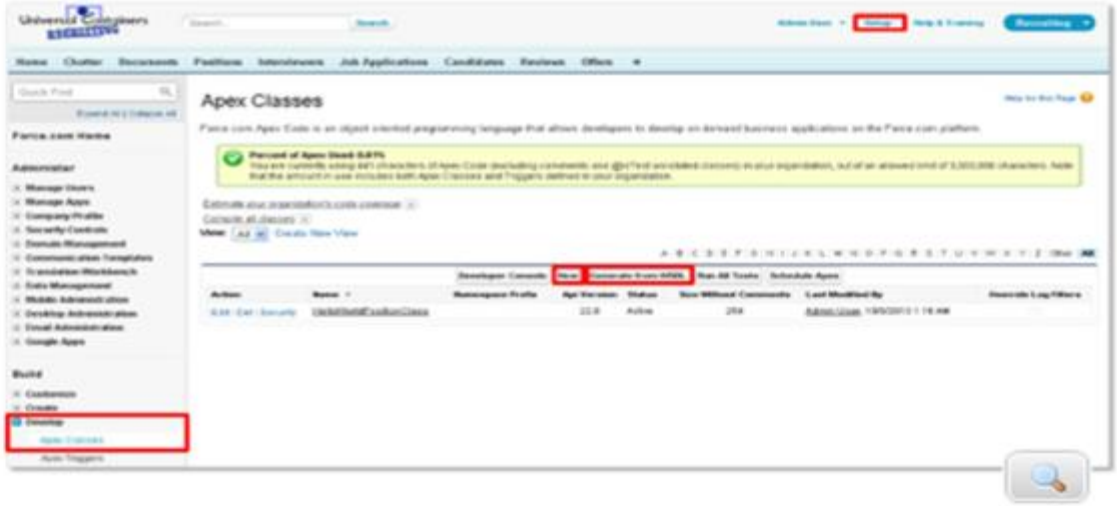
Refer to the Help & Training section in www.salesforce.com to learn about the differences.

Ways to create classes

Using the UI

Using the UI

- Navigate to **Setup | Develop** and **Apex Classes**.
- Click **New** or **Generate from WSDL**.
- Enter the code into the UI or upload a WSDL.



Using the Force.com IDE Project

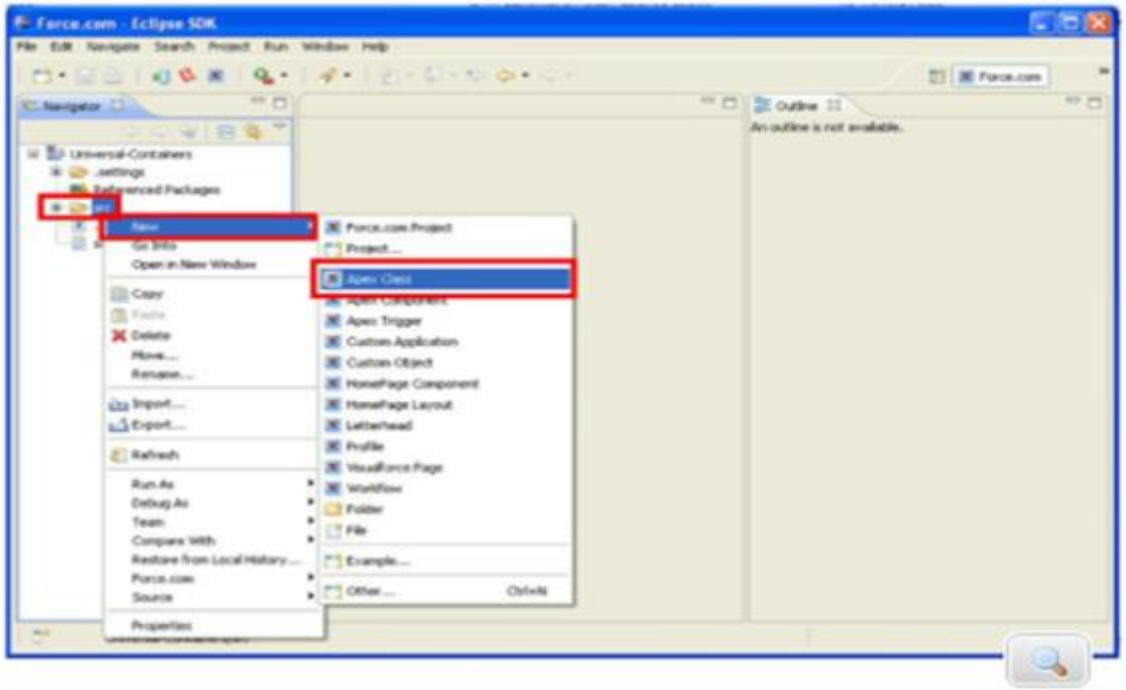
Ways to create classes(Cont.)

Using the UI

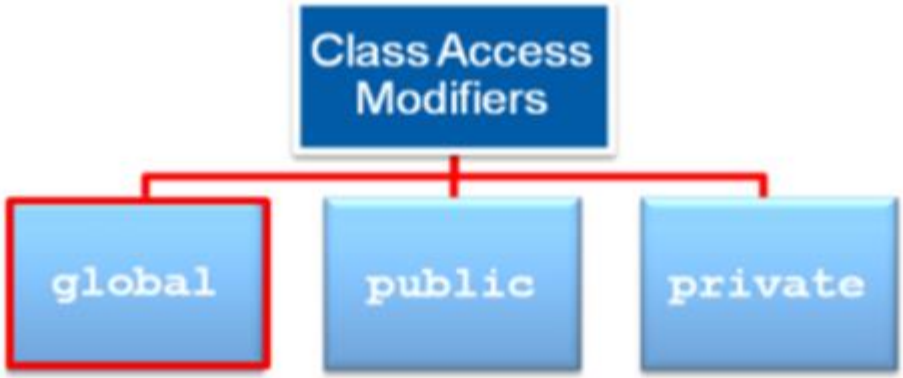
Using the Force.com IDE Project

Using the Force.com IDE Project

- Right-click the **src** folder
- Click **New | Apex Class**.



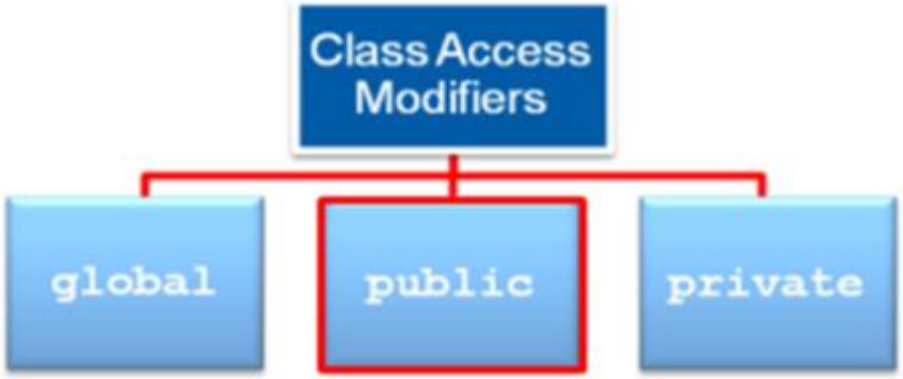
Class Access Modifiers



A global class is accessible by all Apex everywhere:

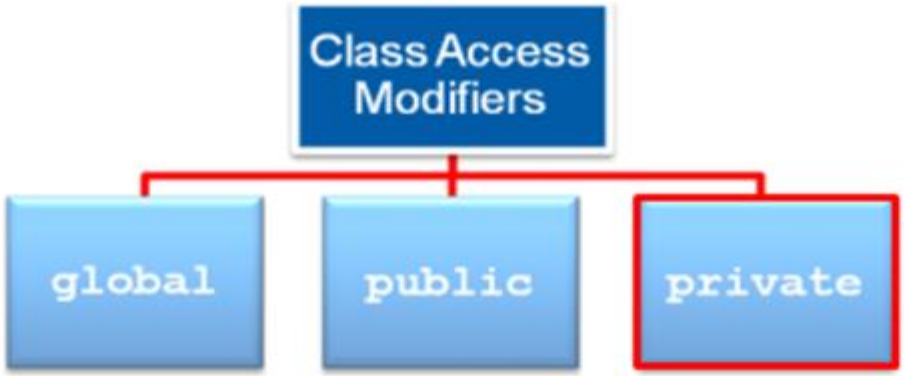
- All methods and attributes with the `webService` keyword must be global.
- All methods, attributes, and inner classes that are global must be within a global class to be accessible.

Class Access Modifiers(Cont.)



- A `public` class is visible across the application, org, or namespace that comprises the class.

Class Access Modifiers(Cont.)



- When you use the `private` keyword in a class definition of an inner class, the inner class is only accessible to the outer class.
- Top-level or outer classes must have either a `global` or a `public` keyword. The default access level for inner classes is `private`.

Interfaces

- Interfaces include only method signatures.
- The methods are not implemented in the interfaces.
- Apex supports both top-level and inner interfaces.

Example of Interface:

```
1A  public interface Manager {
2A      // An interface that defines what a manager looks like in general
3A      Boolean isSeniorExecutive();
4A  }
```

Example of Implementation of the Interface:

```
1B  public class SalesVicePresident implements Manager {
2B      public Boolean isSeniorExecutive(){
3B          return true;
4B      }
5B  }
```

Attributes Syntax

- In a class, attributes and methods define the behavior of objects.
- The syntax for defining attributes and methods is similar to Java.

Syntax of Attribute Declaration

```
modifiers dataType attributeName initialization;
```

Example of Attribute Declaration

```
1  private Integer i, j, k, myInt;
2  String greeting = 'Hello!';
3  public static final Integer MAX_AMOUNT = 200;
```

Apex is not case sensitive. However, it is recommended to use the standard casing similar to Java.

Methods Syntax

Methods can be:

- Defined in anonymous blocks, triggers, or stored classes.
- Overloaded with multiple methods for a class.
- Used wherever System methods are used.

Syntax of Method Declaration

```
modifiers returnDataType methodName(inputParameterlist) {
   //methodCode
}
```

Example of Method Declaration

```
1  public Integer getInt(){
2    return myInt;
3  }
```

Method/Attributes Access Modifiers

Private

Is the default access modifier

A method or attribute defined as “private” can only be accessed within the class it is defined

Is not available for top-level classes

Protected

Attribute or Method are only available to inner classes

Only instance methods and member attributes can use a protected method

Public

Can be used by any APEX code in the app or namespace in which they are defined

Global

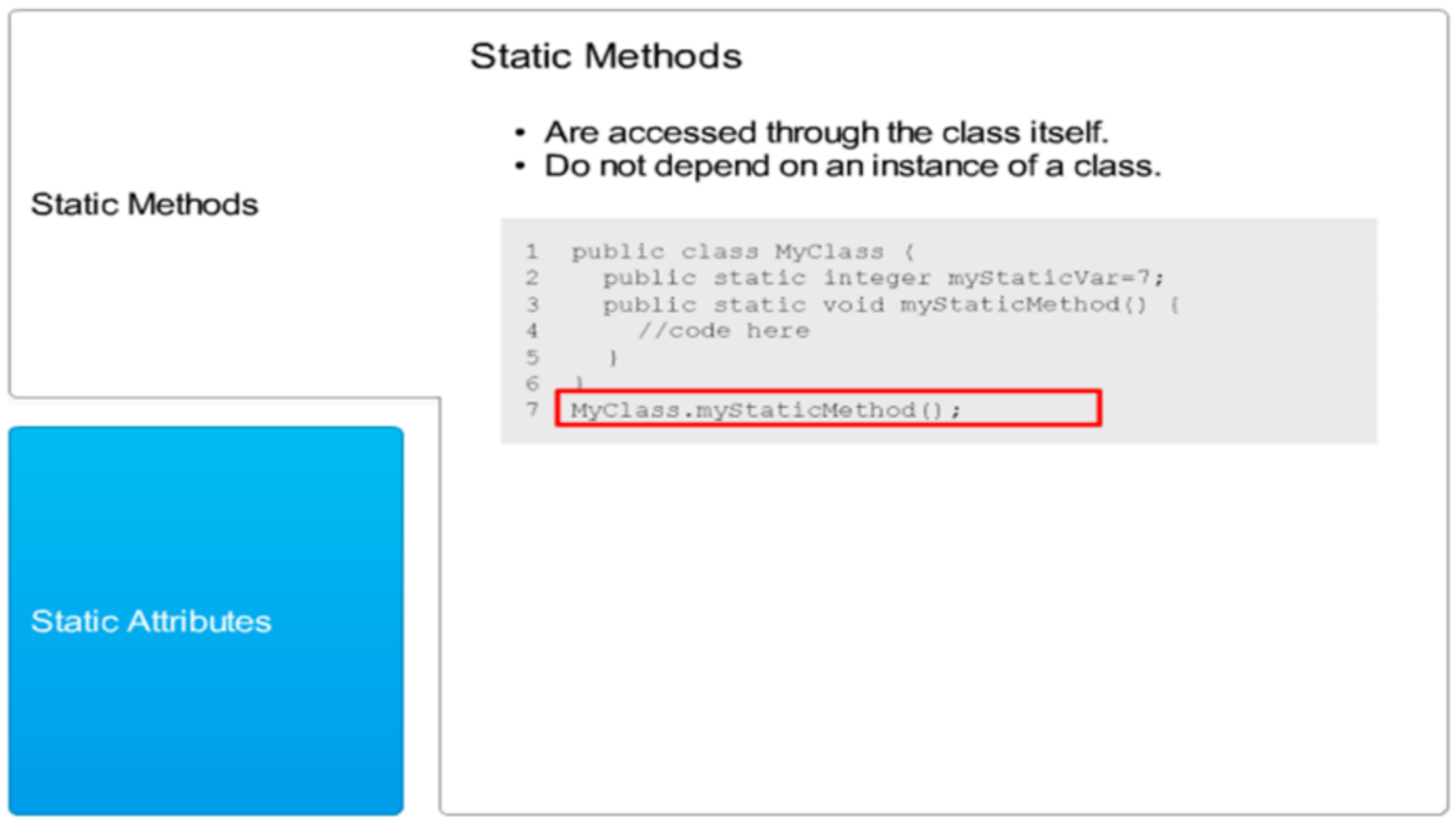
Can be accessed by any APEX code that has access to the class

Can be used across applications

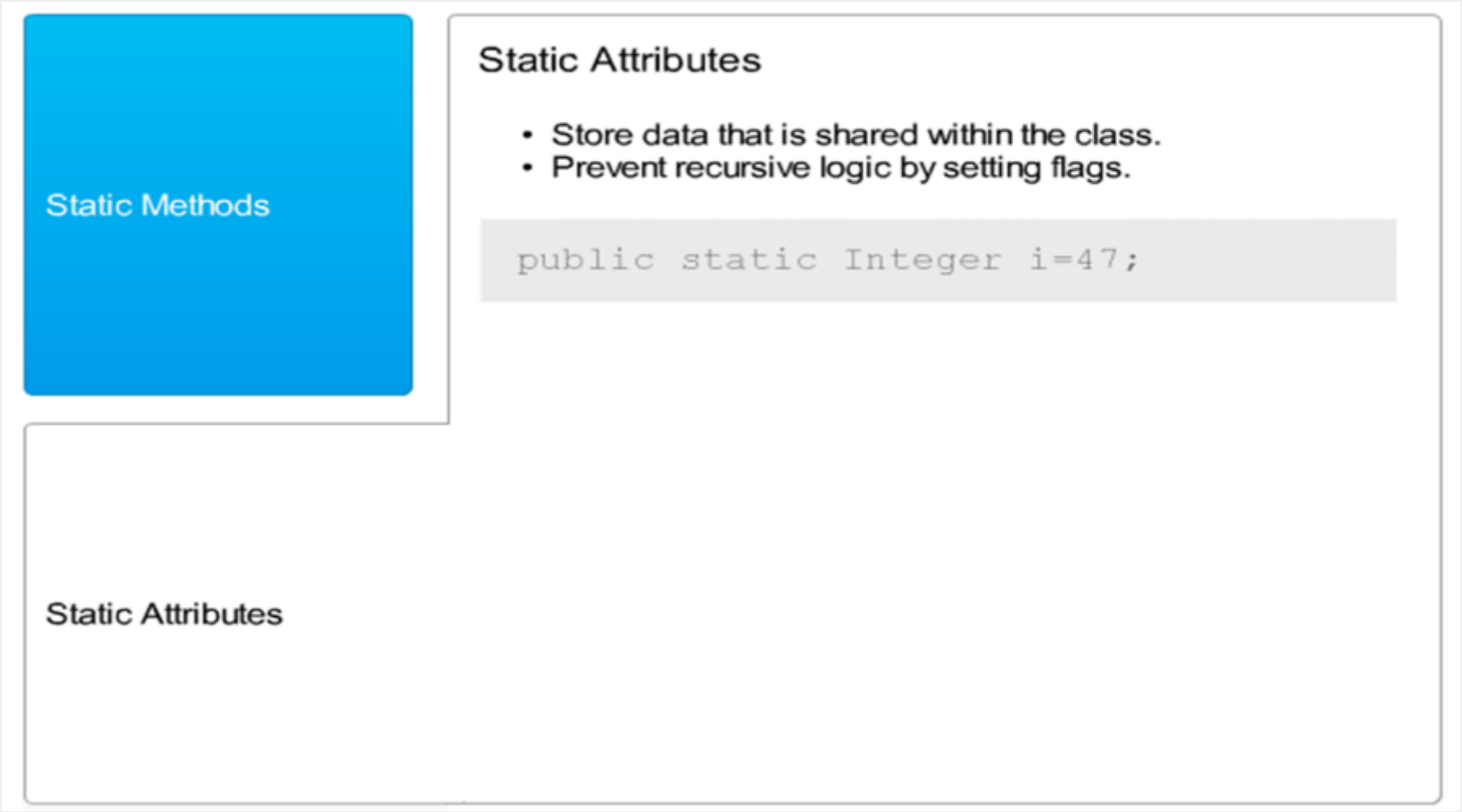
If you use a Global modifier for a method or a variable then the class must be defined as Global

Example Webservices

Static Methods and Attributes



Static Methods and Attributes



Constants

You can

- Assign a value to a constant only once either at the time of declaration or initialization
- Define a constant using both the static and final keywords

Example:

Static final Integer P1_PASSING_SCORE = 41;

Static final Integer P1_NOS_OF_QS = 60;

```
1  public class myClass {
2      static final Integer PRIVATE_INT_CONST;
3      static final Integer PRIVATE_INT_CONST2 = 200;
4      public static Integer calculate() {
5          return 2 + 7;
6      }
7      static {
8          PRIVATE_INT_CONST = calculate();
9      }
10 }
```

Instantiating Objects

Instantiating an object allows you to work with those methods and attributes that are not defined static.

Syntax for Instantiating an Object:

```
ClassName objectName = new Constructor();
```

Example for Instantiating an Object:

```
1  TestObject myObject1 = new TestObject();
2  TestObject myObject2 = new TestObject(3);
3  myObject1.myMethod();
4  myObject2.myVariable = 'Test';
```

Constructors

A constructor:

- Is a special method used to create an object of a class.
- Has the same name as the class.
- Is the first method that is invoked in the class.
- Does not have an explicit return type.
- Is available by default in each class as invisible and without parameters.

Example: Overloaded Class Constructor

```
1  public class TestObject {
2    private static final Integer DEFAULT_SIZE = 10;
3    Integer size;
4    //Constructor with no arguments
5    public TestObject() {
6      this(DEFAULT_SIZE); // Using this(...) calls the one arg constructor
7    }
8    // Constructor with one argument
9    public TestObject(Integer ObjectSize) {
10     size = ObjectSize;
11   }
12 }
13 TestObject myObject1 = new TestObject(42);
14 TestObject myObject2 = new TestObject();
```

Apex System-Delivered Classes

System Class

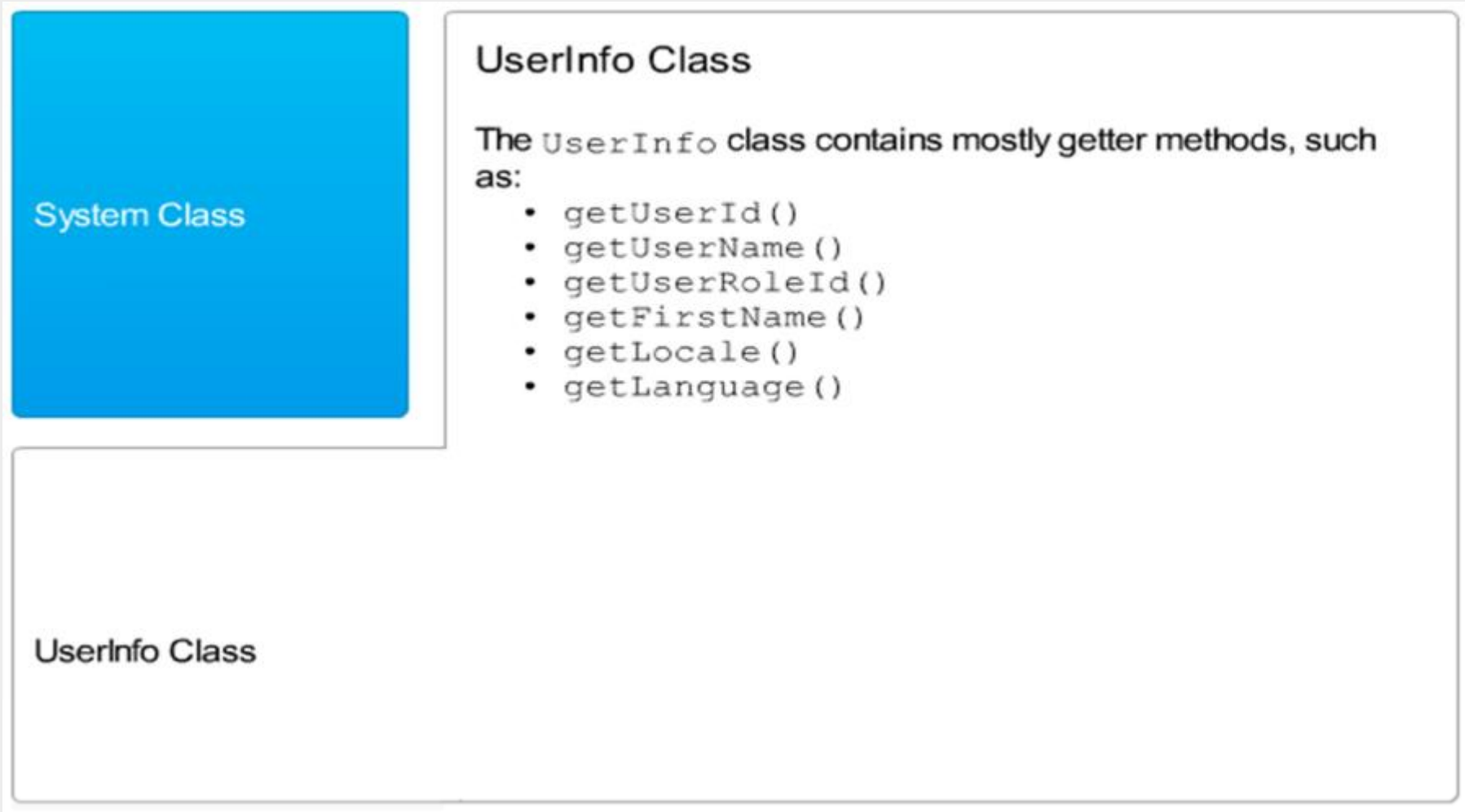
UserInfo Class

System Class

- Is a static class.
- Contains only static methods, which can be directly called with no instantiation.
- Includes the following methods:
  - `debug()`
  - `now()`
  - `today()`
  - `assert()`
  - `assertEquals()`
  - `assertNotEquals()`

System Static Methods	Description
now	Returns the current time.
today	Returns the current date in the current user's time zone.
assert	Asserts that a condition is true. If the condition is false it throws a runtime exception with an optional message.
assertEquals	Similar to assert, but compares two arguments if they are equal.
assertNotEquals	Similar to assert, but compares two arguments if they are unequal.
debug	Writes the message argument in string format to the execution debug log.

Apex System-Delivered Classes



Governor Limits

Why is this important?

APEX runs in an multi-tenant environment

It is important to ensure that runaway APEX code or processes do not monopolize share resources

The APEX engine manages memory, database resources, etc.

Will terminate process when limits are exceeded

Are typically reset per transaction

Limits can be found here

Limit Methods

- Apex runtime engine strictly enforces a number of limits on resources.
- Limit methods can be used for debugging and printing messages.
- There are two versions of each built-in method:
  - The first version returns the amount of resource being used.
  - The second version returns the total amount of the resource available.

Methods	Description
`getDMLRows`	Returns the number of records processed with any DML statement.
`getDMLStatements`	Returns the number of DML statements processed.
`getHeapSize`	Returns the amount of memory in bytes used for the heap.
`getQueries`	Returns the number of SOQL queries issued.
`getQueryRows`	Returns the number of records returned by SOQL queries.
`getScriptStatements`	Returns the number of statements executed.

Class Keyword: Sharing

With Sharing

If you apply "with sharing" to a class, record level sharing privileges are enforced to the class

When performing DML operations the user can only update the records to which they have edit-level access

Anonymous blocks executes in this mode

Without Sharing

Also known as System Mode

All APEX code runs in this mode (except Anonymous Block)

This ignores all CRUD permissions on object, field-level security, and record sharing priviliges

When not specified "Without Sharing" is used

If a class is called from a class that does not have sharing enforced, sharing is not enforced for the called class as well

Manual Sharing Vs Apex Sharing

Manual Sharing:

Manual sharing occurs when a user clicks the Sharing button to manually share a record

The reason for the sharing is deplayed in the user interface is typically "Manual Sharing"

APEX Sharing:

Is a way to create custom sharing reasons and control access levels for those reasons using custom Apex scripts

Can be created only on custom object configuration menu

Hands-On Exercise

Apex: Data Types and Logic Exercise Guide

Exercise 2-1: Writing Basic Anonymous Blocks with Apex Data Types

https://lms.cfs-api.com/v1/content/8f644ab0-76e6-4f49-9d28-209b1b98afdd/presentation_content/external_files/DataTypesandLogicExerciseguide.pdf

Apex Workbook:

https://resources.docs.salesforce.com/sfdc/pdf/apex_workbook.pdf