Learning Goals

Click to continue

This session will help learners to understand the following topics,

- Java Class
- Java packages

Java Classes

Click to continue

Classes are the core components which are used to build a java program. The logic of the program are implemented in the java class.

An java application can be built using one or more java classes, each class contains application business logics.

A Quick Recap: Objects has ***state*** & ***behavior***.

- ***States*** (**viz:** ***Data***) are represented by the variables of the classes.
- ***Behavior*** (**viz:** ***Business logic***) are implemented as methods in classes.

IMPORTANT NOTE:

- A physical .java file can have more than one classes.
- The java file should be named after the class which is declared public.
- There cannot be two public classes defined in the same java file.

Illustration: Java Class

Click to continue

```
class Student

{
      int studentAge=0;
      long studentId=75;
      long studentFees=0;
      void calculateFees()
            {
                  // logics of calculating fees
                  //calculation goes in here
                   studentAge= 20;
                   long fees = 1000*4;
                   system.out.println(“Fees =“+fees);
            }
     }
```

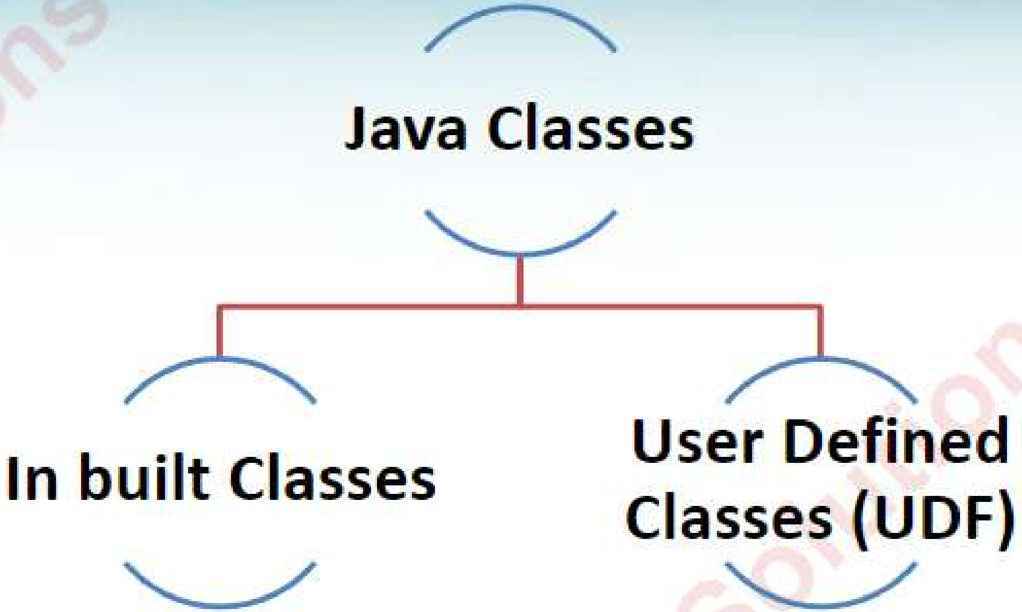
Three variables student age, fees and student id are declared in student class.

The method sets the age as 20 calculates the fees and prints the fee.

Important Note: Java is case sensitive. For example, “**Student**” and “**student**“ are two different classes.

Java Classes Types

Click to continue



Java Classes

In built Classes

User Defined Classes (UDF)

The classes provided by the java runtime environment are referred to as **inbuilt classes**.

Example: String, Integer.

The java classes developed by the user are referred top as **User defined classes.**

Java packages

Click to continue

Packages are namespaces used for logically grouping the classes together into a single unit.

The java classes are logically grouped into packages.

Why packages?
A shopkeeper was selling two varieties of oranges one from Italy and other from Portugal. He found it very tough to differentiate the oranges from both the countries as they looked very identical.

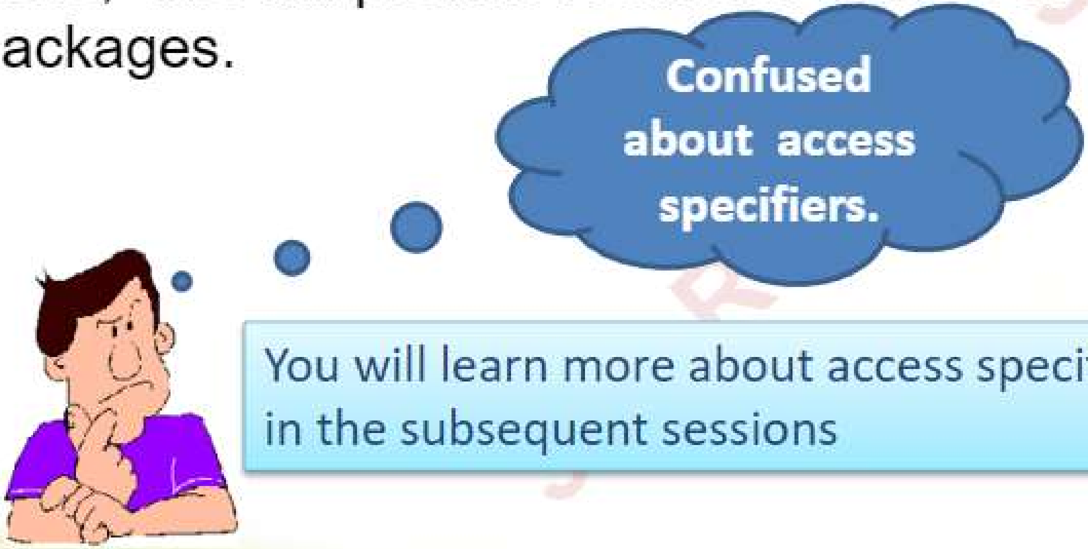
How did he solve this problem?
He placed the oranges in separate baskets and labeled the baskets.

Similarly Java classes can be logically grouped inside packages. In case there are two classes with the same name they can be identified by their package where they reside.

Packages advantages:

Click to continue

- ✓ **Packages** groups classes based on their functionality which eases maintainability.
- ✓ Two classes can be declared with same name under different packages.
- ✓ Packages provide a level of security by declaring the classes with ***access specifiers***, this will provide restricted access to classes from outside other packages.



You will learn more about access specifiers in the subsequent sessions

Packages are used logically to group the java classes by which the application can be made easily maintainable and organized.

“package” is the key word used for creating packages.

How to create package:

package <package name>;

Example: package com.ripples

Important: When a class within a package is compiled it will create folders similar to the package name.

Illustration: File Student.java under com.ripples package on compilation will generate a Student.class under com→ripples folder.

Import classes

Click to continue

When developing a class if you want to reference a class we need to use "**import**" keyword to import the class.

The **import** statement is used to bring certain classes, or entire packages, into visibility.

How it is done?

import <package-name>.<class-name>;

(or)

import <package-name>.*;

The first option imports a specific class into the class. Second option imports all the class in a specific package.

Example: **import com.ripples.Employee;** (or)

import com.ripples.*

Package Illustration

Click to continue

```
package com.ripples;
class Employee{
 void displayEmployeeID()
   {
     System.out.println(“Employee ID”);
   }
 }
```

```
package  com.ripples.company;
import com.ripples.Employee;
class Company{
    public static void main(String args[])
  {
     Employee employee = new Employee();
     employee.displayEmployeeID();
  }
}
```

NOTE: One can also import all the classes as **import com.ripples.*.** **But this is not a good practice.**

How to create objects?

Click to continue

The following are the steps by which objects are created ,

- **Declare -** Create a reference to store the object instance.
- **Instantiation -** Use the new keyword to create an object.
- **Initialization -** The object values can be initialized using a constructor.

Object creation illustration

Click to continue

The program creates a student object and invokes the calculateMarks method.

```
class StudentExecute
{
        public static void  main (String args[])
        {
           Student obj =null;

            obj = new Student();

           obj.name = “Ram”;

           obj.calculateMarks();
        }
   }
```

Declares a Studentobject.

Instantiates and initialize the student object using constructor.

Sets the student name.

Triggers the calculate marks method in the student object.

Try it out - Assignment

Click to continue

1. Create a java class “Employee” add a integer variable “employeeId”.
2. Create a method “displayEmployeeId” which will print the employee id of the employee in the format

   “The employee id is <employee Id>”

3. Create a java class “EmployeeMain” add a main method which will
   - Create a object instance of the Employee class
   - Set the employee id to value “2345”
   - Invoke the method ““displayEmployeeId””.
4. The message needs to be displayed in the console.

Expected Output: “The employee id is 2345”