# Eclipse

Session 8

# What's Eclipse??

# Eclipse

**Eclipse** is an integrated development environment (IDE) used in computer programming, and is the most widely used Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins, including Ada, ABAP, C, C++, C#, COBOL, etc.

# Check for Java on Your Computer

To verify that Java is already installed on your computer:

- Windows or Linux operating systems:

– Enter java -version in a command window.

- Mac operating system:

– Use the Software Update option from the Apple menu.

# Getting Started With Eclipse

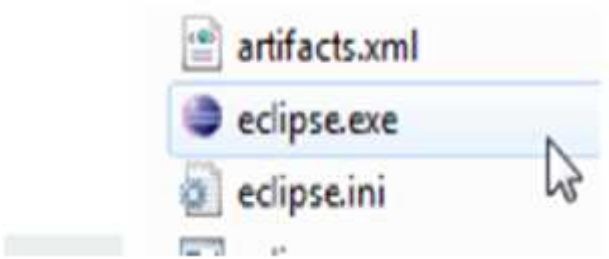
# Lesson Objectives

In this lesson you will learn to:

- Identify components of Eclipse
- Identify components of a Java application
- Compile an application
- Test to ensure application is complete

# Launch Eclipse

- On a computer with Windows double-click on the file eclipse.exe. On a Linux or Mac computer double click on the file eclipse.
- When prompted, enter the pathname for the workspace into which you will store your Java projects and click the OK button. This can be your c:\ drive, or possibly a network drive.
- Eclipse will start and display the Welcome page.
- Close the Welcome page by clicking the X next to the Welcome tab name.

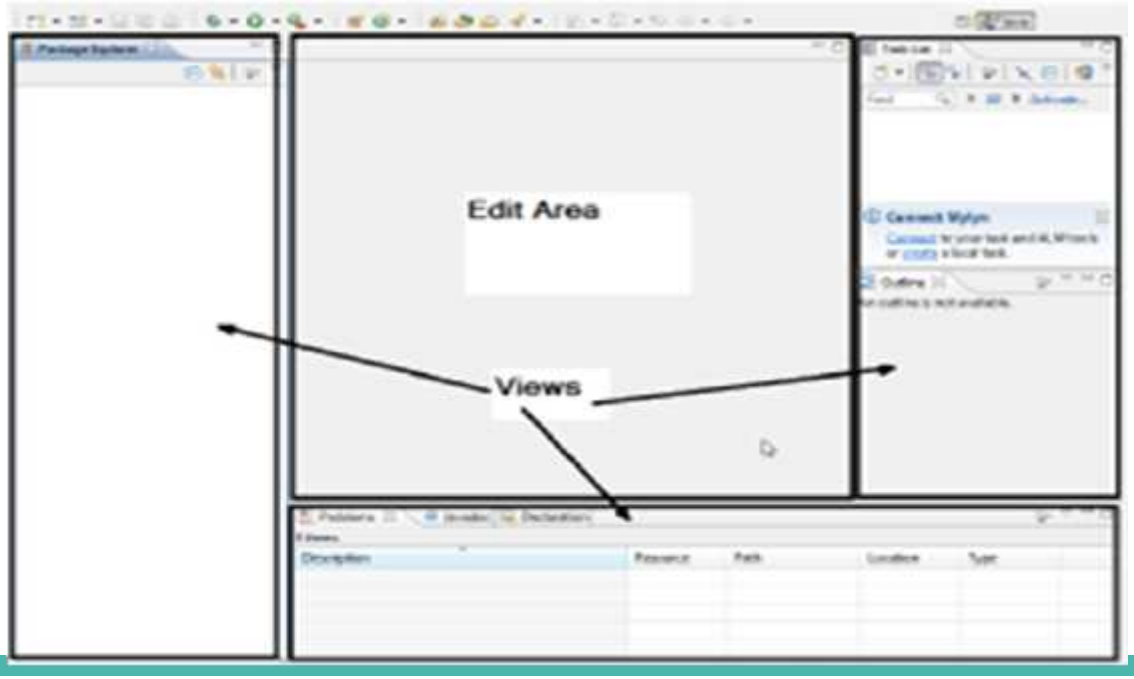

artifacts.xml
eclipse.exe
eclipse.ini


eclipse


Welcome to Eclipse

# Eclipse Component

- Eclipse provides an edit area and several views.
- An editor is where you type in your Java source code.
- Views are sub-windows that provide information about your project.



# Step by Step to Create a Program in Eclipse

- Create a Project.
- Create a Package (inside the src folder of the project).
- Create Class(es) inside the package.
- Compile the Java code. This creates a .class file.
- Run the Java code from the Driver class.

# Project 1

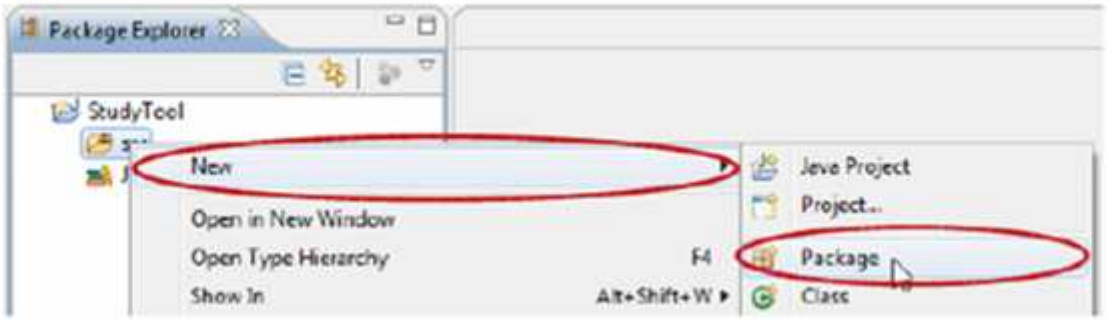
# Create a StudyTool Project

- Choose File → New → Java Project*.
- Enter a project name and click Finish.
- All project values are set to default by clicking the Finish button.

# Create a studyTool Package

- Select the src folder in the Package view.
- Right click the src folder and choose New → Package.

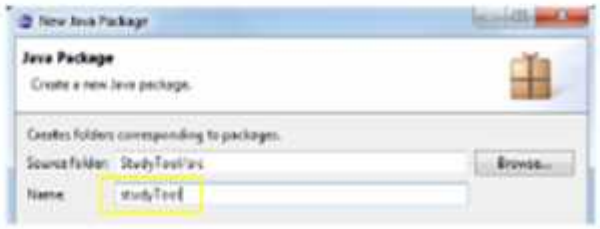
A package, in Java, is a mechanism for organizing Java classes into namespaces or containers. In Eclipse, packages are created inside projects.

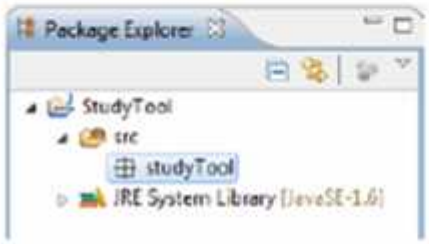


# Naming Packages

- Name this package the same name as the Project using lower camel case.

> Camel case is the practice of stringingCapitalizedWords together with no spaces. Lower camel case does not capitalize the lead word.

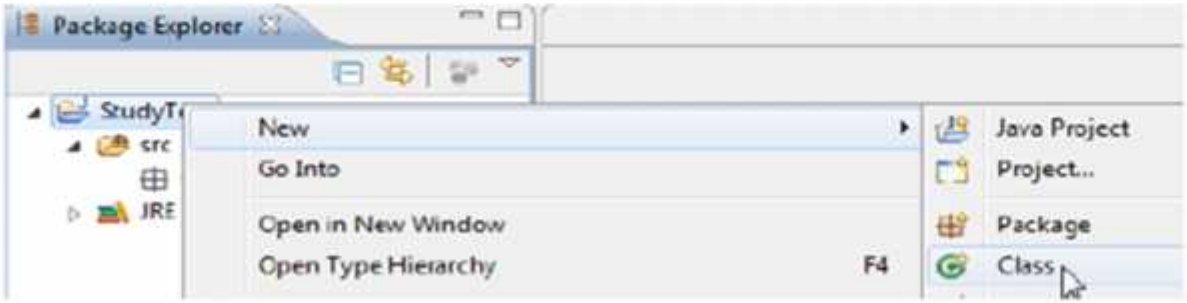


# Create StudyPage Class

1. Right click on the project name to create a new class named StudyPage in the studyTool package.
2. Select the option to create the main method.

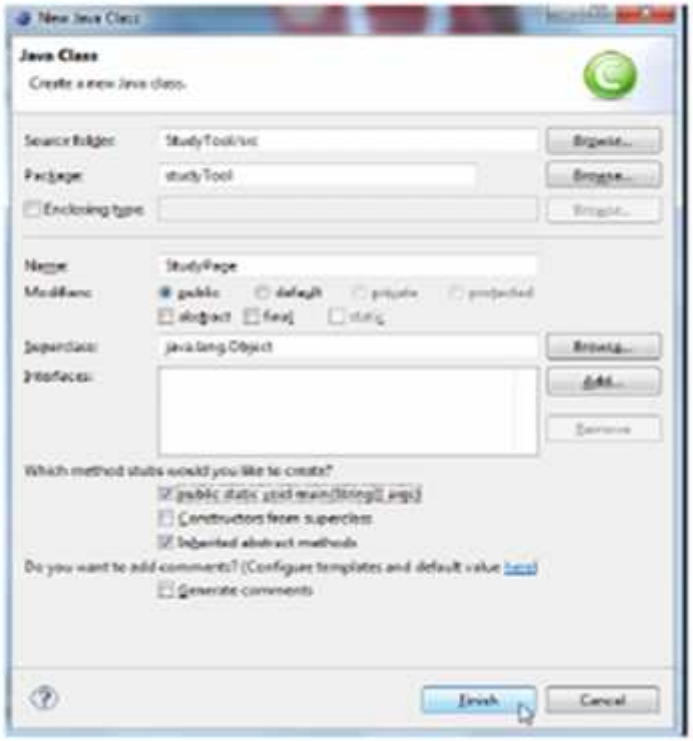
A class in Java is a construct that is used as a blueprint to create objects. It can also be a construct in which objects are created.



# Class Display

- The option is selected to create the main method.

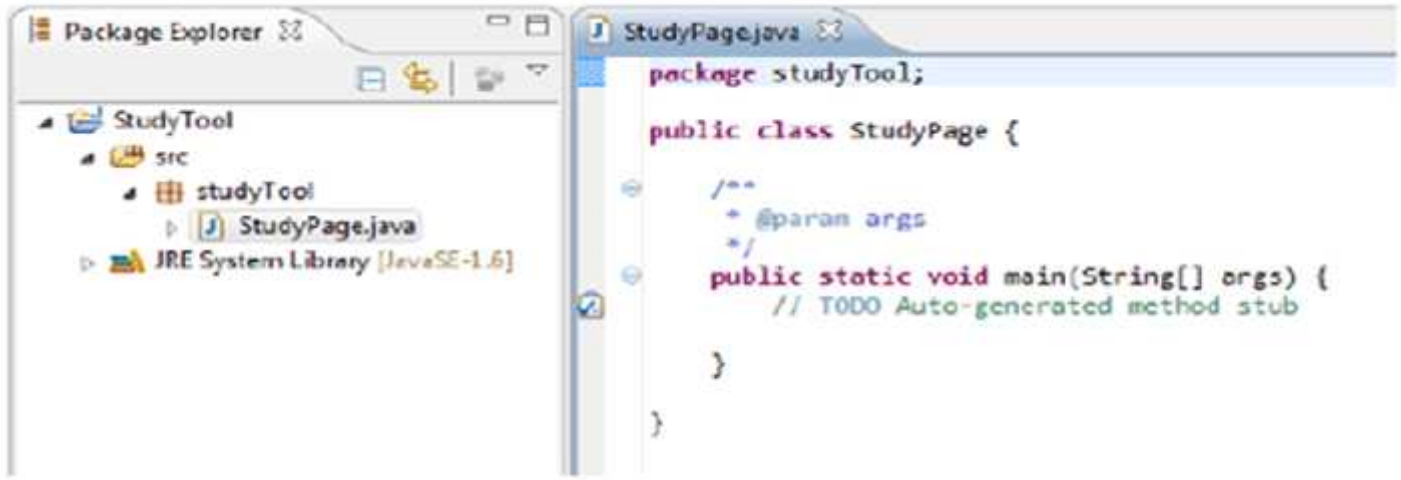
A main method, in Java, is the method inside a class that runs when the class is compiled and run. This class is referred to as the Driver Class.



# New Class

Notice the following:

- The StudyPage.java class appears in the studyTool package in the Package Explorer View.
- The StudyPage.java class is created with the main method.



# Add a Code for the Program

```
package studyTool;
import java.util.Scanner;

public class StudyPage {
   public static void main(String[] args) {

           Scanner scanterm = new Scanner(System.in);
           String termvar;
           System.out.println("Enter a study term");
           termvar = scanterm.nextLine();
           Scanner scandef = new Scanner(System.in);
           String termdef;
           System.out.println("Enter a definition");
           termdef = scandef.nextLine();

           System.out.println(termvar + ":   " + termdef);
     }
}
```

# Compile and Run Java Code

- Right click on class.java in the package view.
- Choose Run As → Java Application.
- Save the class when prompted.
- Note results display in the Console View.

# Save Your Project

# PROJECT 2

# Object and Driver Class

# Lesson Objectives

In this lesson you will learn to:

- Create an Object class
- Create a Driver class

# General Java Program Form

## Driver classes:

Contain a main method.

- A main method is necessary to run a Java program in Eclipse.

The main method may include:

- Instances of objects from an object class
- Variables
- Loops, conditional statements (if else)
- Other programming logic

## Object classes

- Are classes that define objects to be used in a driver class.
- Can be found in the Java API, or created by you.

Examples: String, BankAccount, Student, Rectangle

# Java API

The Java API is a library of packages and object classes that are already written and are available for use in your programs.

The Java API documentation is located here:
– http://docs.oracle.com/javase/8/docs/api/

# Example Java API : String Class

The constructor that is most common for this class is:

String(String original)

A String is an object that contains a sequence of characters. Declaring and instantiating a String is much like any other object variable.

Common Methods include:

| Method | Description |
|---|---|
| charAt(int index) | Returns the char value at the specified index. |
| length() | Returns the length of this string. |
| substring(int beginIndex) | Returns a new string that is a substring of this string. |

# Create Object Class

- All Java classes are created in a text file with the same name as the class.
- These files also have a .java extension.

The following is example syntax to create a programmer-created object class.

- The Java keywords are:
  - package (optional)
  - import (optional)
  - public class

```
package <package_name>;
import <other_packages>;

public class ClassName
    {
        <variables (also known as fields)>;
        <constructor method(s)>;
        <other methods>;
    }
```

# Key Term

| Term | Definition |
| --- | --- |
| Package keyword | • Defines where this class lives relative to other classes, and provides a level of access control.<br>• Use access modifiers (such as public and private) to control access. |
| Import keyword | • Defines other classes or groups of classes that you are using in your class.<br>• The import statement provides the compiler information that identifies outside classes used within the current class. |
| Class keyword | • Precedes the name of the class.<br>• The name of the class and the file name must match when the class is declared public (which is a good practice). However, the keyword public in front of the class keyword is a modifier and is not required. |
| Class variable or instance fields (often shortened to fieds) | Variables, or the data associated with programs (such as integers, strings, arrays, and references to other objects). |

| Term | Definition |
| --- | --- |
| Constructors | Methods called during the creation (instantiation) of an object (a representation in memory of a Java class.) |
| Methods | Methods that can be performed on an object. They are also referred to as instance methods. Methods may return an object's variable values (sometimes |

# Java Keyword

A Java keyword is a word that has a special function in the Java language, and cannot be used as names for classes, methods, or variables.

- All Java programs use Java keywords.
- Examples include the following words: class, public, String, int, for, while, and double.
- The font color for Keywords will change in the Eclipse editor.
- List of Java keywords:

http://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

# Import Keyword

The import keyword is used to identify packages or object classes that you want to use in your class.

- You can import a single class or an entire package.
- You can include multiple import statements.

Import statements follow the package declaration and precede the class declaration.

An import statement is not required, and by default, your class always imports java.lang from the API.

- –http://docs.oracle.com/javase/8/docs/api/java/lang/Package.html

Additional examples of import statements:

```
import java.util.Date;        // Import the individual class.

import java.util.*;           //Import the entire package.

import java.util.Date;        //Import using multiple statements.
import java.util.Calendar;
```

# Naming Convention ( Camel Case )

Camel case is the practice of stringing capitalized words together with no spaces.

- Lower camel case strings capitalized words together but the lead word is not capitalized.

  For example: thisIsLowerCamelCase.

- Upper camel case strings capitalized words together, but the lead word is capitalized.

  For example: ThisIsUpperCamelCase.

# Let's Code Program

# Create Program

We will create a model, or programmatic representation, of a Student.

1. Create New Project, name as DataStudentProject
2. Create New Object Class, name as Student
   The Student class will contain variables for student Id type as Integer, name as String, social security number(ssn) as String, grade point average(GPA) as double, and school code as Integer.
1. Create New Driver class, name as StudentTester.
2. Save your project

# Constructor Method

A constructor method is unique in Java because:

- The method creates an instance of the class.
- Constructors always have the same name as the class and do not declare a return type.

With constructor methods:

- You can declare more than one constructor in a class declaration.
- You do not have to declare a constructor, in fact, Java will provide a default (blank) constructor for you.
- If you declare one or more constructors, Java will not provide a default constructor.

# Parameter In Constructor

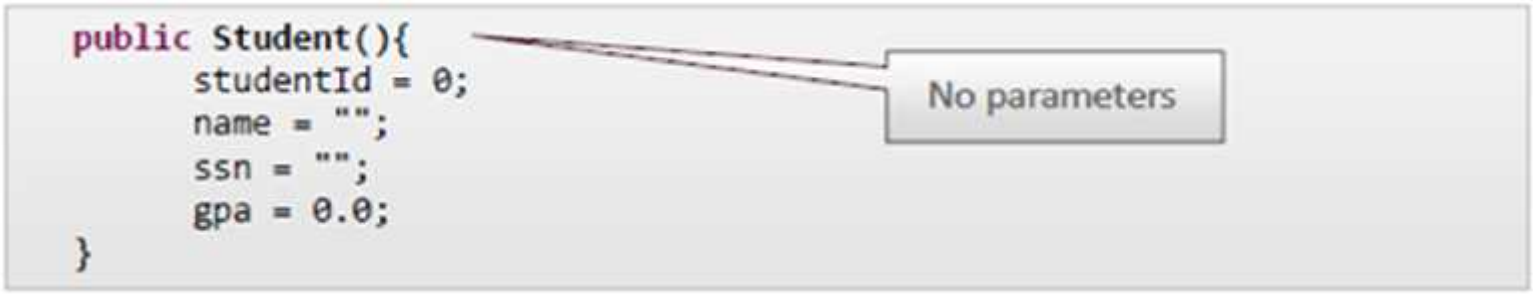
- If you create a constructor without parameters (the parenthesis is empty), you can leave the contents of the constructor (between the { and }) blank.

This is called a default constructor, and is the same as the Java-provided constructor if you do not declare one.


- If you create a constructor with parameters (the parenthesis is NOT empty), you would also initialize the variables between the { and }. Constructor will initialize the class variables with the values that are sent in from the main driver class.

# Let's Code Program

- If you create a constructor without parameters (the parenthesis is empty), you can also initialize the variables between the { and }. This is also called a default constructor.
- This constructor initializes the numeric class variables to zero, and object variables (such as Strings) to null. It has the same results as the previous slide, but the values are more evident.



# Let's Code Program

- Add constructor method with parameter in object Class (Student).
  The Constructor include parameter student Id, name, social security number(ssn), grade point average(GPA).

```
public Student(int x, String n, String s, double g){
        studentId = x;
        name = n;
        ssn = s;
        gpa = g;
}
```

# Accesor and Mutator Method

It is common to create a set of methods that manipulate and retrieve class data values:

- Accessors (getters): Methods that return (get) the value of each class variable.
- Mutators (setters): Methods that change (set) the value of each class variable.

# Let's Code Program

- In Student class, Add the getStudentId() method and the setStudentId() method as shown below.

```
public int getStudentId()
      {
         return studentId;
      }
public void setStudentId(int studentId)
      {
         this.studentId = studentId;
      }
```

- Add another Accessor methods in Student Class
  getName(), getSSN(), getGPA()
- Add another Mutator methods in Student Class
  setStudentId(), setName(), setSSN(), setGPA()

# Let's Code Program

- Add the toString() method to the Student object class.

Note, any String object can be built by you to display the information about each student:

```
public String toString()
       {
          String s1 = "";
          s1 =  "Student Id: " + getStudentId() +
                   "Student Name: " + getName() +
                   "Student SSN: " + getSSN() +
                   "Student GPA: " + getGPA();
          return s1;
       }
```

- Compile the java code and save your Project

- Then, create a StudentTester Driver Class main Method as follows:

```
public class StudentTester
{
   public static void main(String args[]
   {
      Student s1 = new Student(123, "Mary Smith", "999-99-9999", 3.4);
      System.out.println(s1);
      Student s2 = new Student();
      s2.setStudentId(124);
      s2.setStudentName("John Jacoby");
      s2.setSSN("123-45-6789");
      s2.setGPA(4.0);
      System.out.println(s2);
      Student s3 = new Student();
      System.out.println(s3);
   }
}
```

- Compile and Run Student Tester class.

# Save Your Project

# Practice

- Create a project, named Shape
- Create a driver class, named ShapePage
- Create object classes, named Square, Rectangle, Circle, Triangle, Rhombus, and Parallelogram
  - Add Constructor, Accesor and Mutator for every variable needed in the object classes
  - Calculate the area of all shape.
  - Display it, just like Project 2
- Save the project