Java Package

What is Package

* A package can be defined as a group of similar types of classes, sub-classes, interfaces enumerations. While using packages it becomes easier to locate or find the related classes and packages provides a good structure or outline of the project with a huge amount of classes and files.

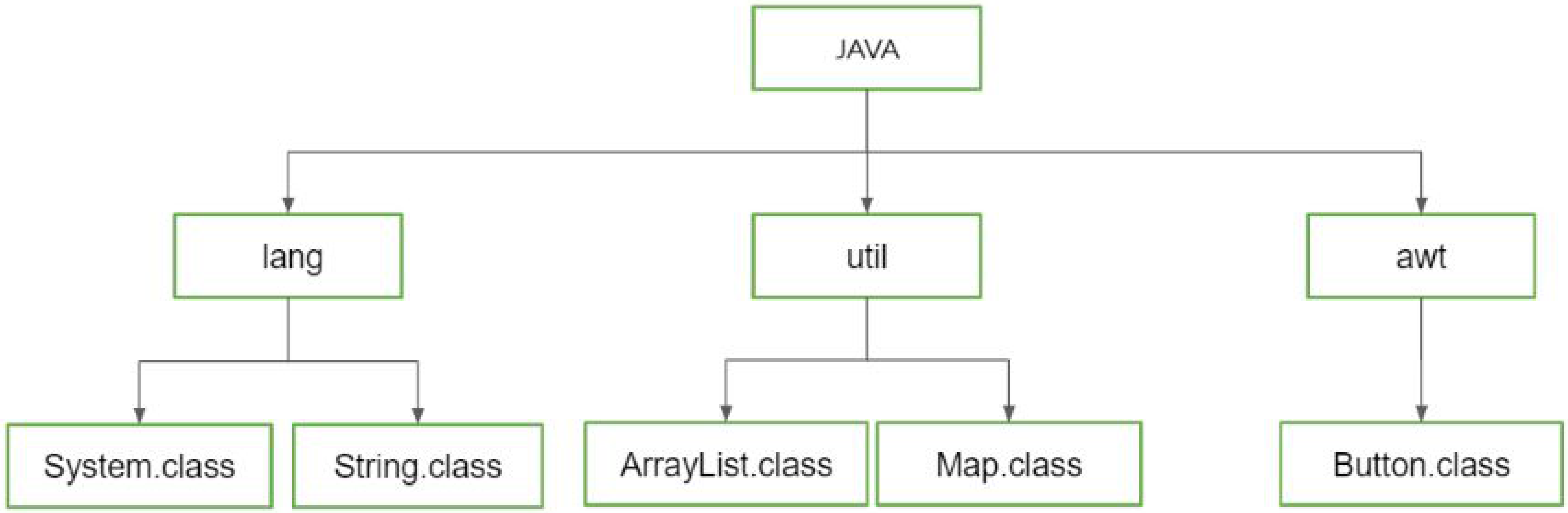
Advantage of Java Package

- 1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.
- 2) Java package provides access protection.
- 3) Java package removes naming collision.

Packages are divided into two parts:

- **Built-in packages:** In java, we already have various pre-defined packages and these packages contain large numbers of classes and interfaces that we used in java are known as Built-in packages.
- **User-defined packages:** As the name suggests user-defined packages are a package that is defined by the user or programmer.

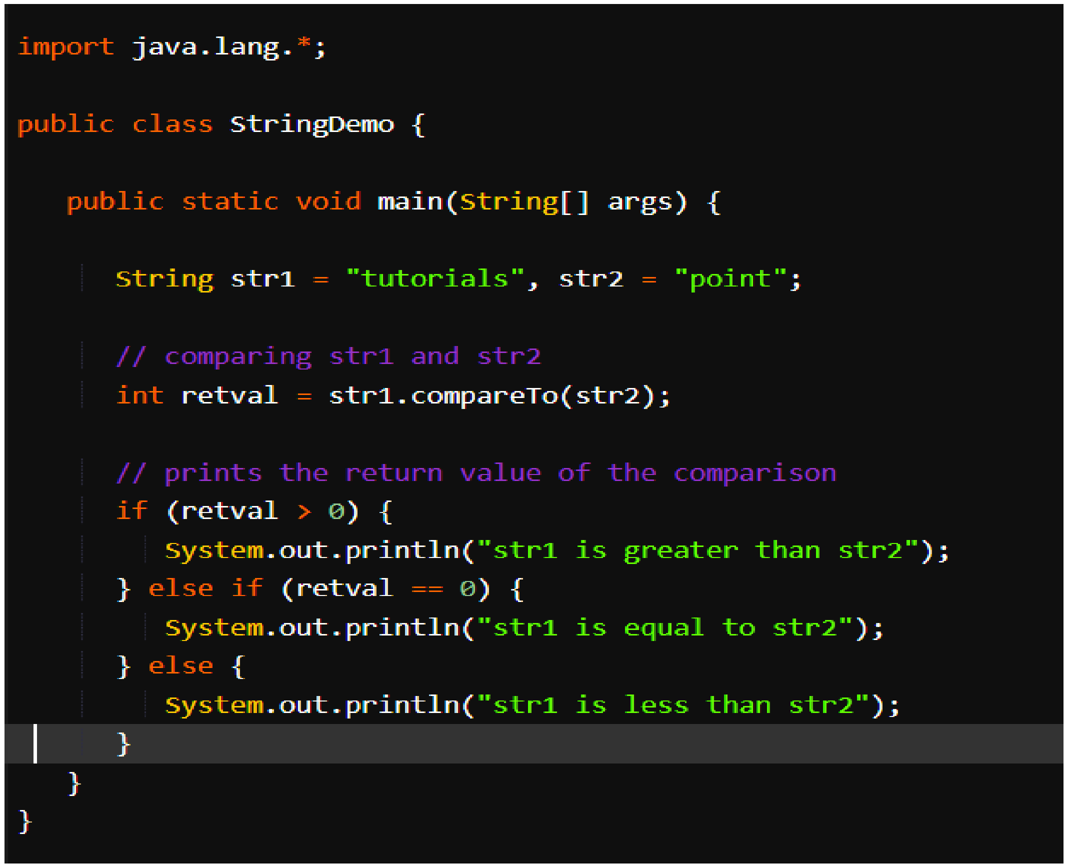
Built-in packages



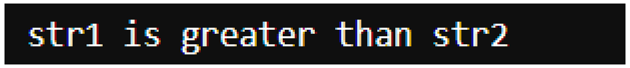
Examples of Built-in Packages

- **java.sql**: Provides the classes for accessing and processing data stored in a database. Classes like Connection, DriverManager, PreparedStatement, ResultSet, Statement, etc. are part of this package.
- **java.lang**: Contains classes and interfaces that are fundamental to the design of the Java programming language. Classes like String, StringBuffer, System, Math, Integer, etc. are part of this package.
- **java.util**: Contains the collections framework, some internationalization support classes, properties, random number generation classes. Classes like ArrayList, LinkedList, HashMap, Calendar, Date, Time Zone, etc. are part of this package.
- **java.net**: Provides classes for implementing networking applications. Classes like Authenticator, HTTP Cookie, Socket, URL, URLConnection, URLEncoder, URLDecoder, etc. are part of this package.
- **java.io**: Provides classes for system input/output operations. Classes like BufferedReader, BufferedWriter, File, InputStream, OutputStream, PrintStream, Serializable, etc. are part of this package.
- **java.awt**: Contains classes for creating user interfaces and for painting graphics and images. Classes like Button, Color, Event, Font, Graphics, Image, etc. are part of this package.

Example: Java.lang

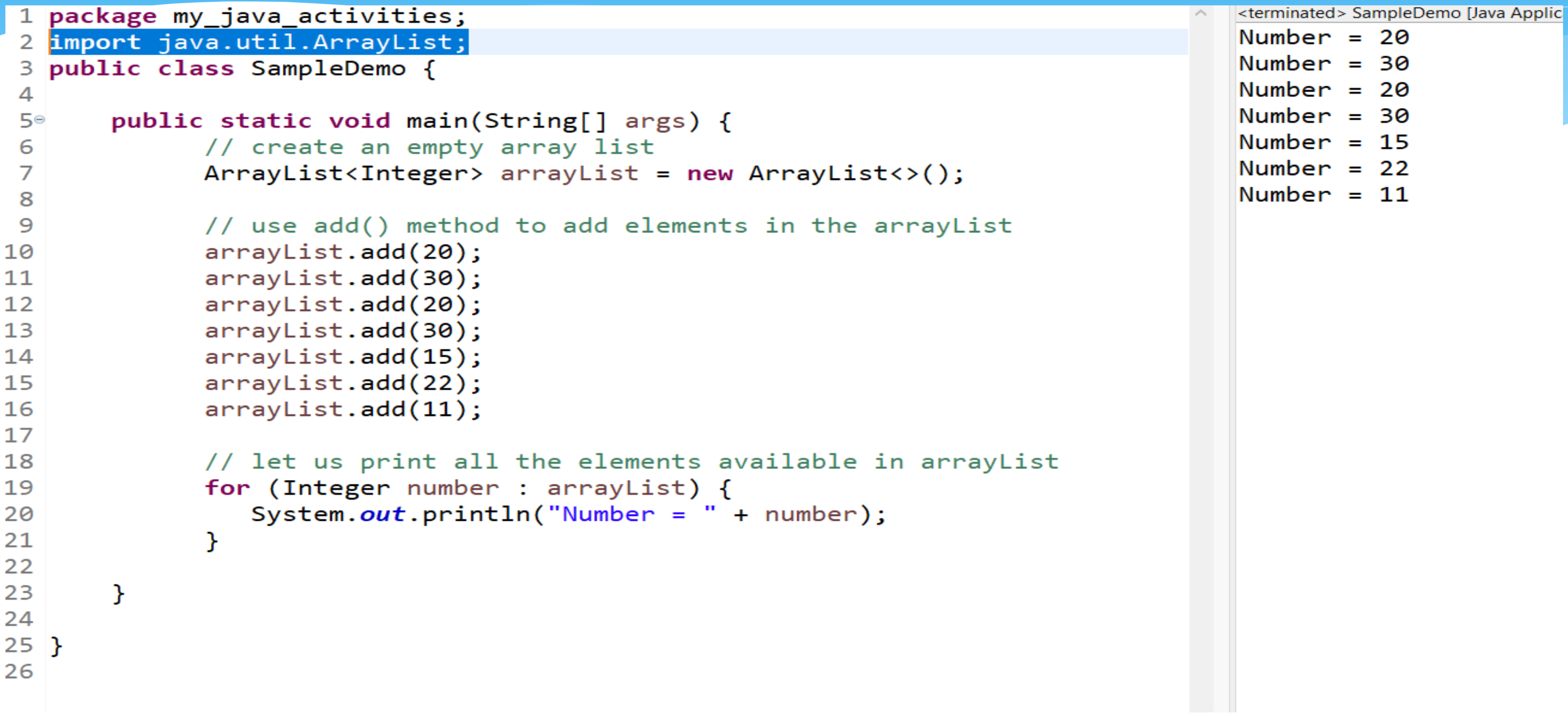


```java
import java.lang.*;

public class StringDemo {

   public static void main(String[] args) {

      String str1 = "tutorials", str2 = "point";

      // comparing str1 and str2
      int retval = str1.compareTo(str2);

      // prints the return value of the comparison
      if (retval > 0) {
         System.out.println("str1 is greater than str2");
      } else if (retval == 0) {
         System.out.println("str1 is equal to str2");
      } else {
         System.out.println("str1 is less than str2");
      }
   }
}
```



```
str1 is greater than str2
```

Example: Java.util

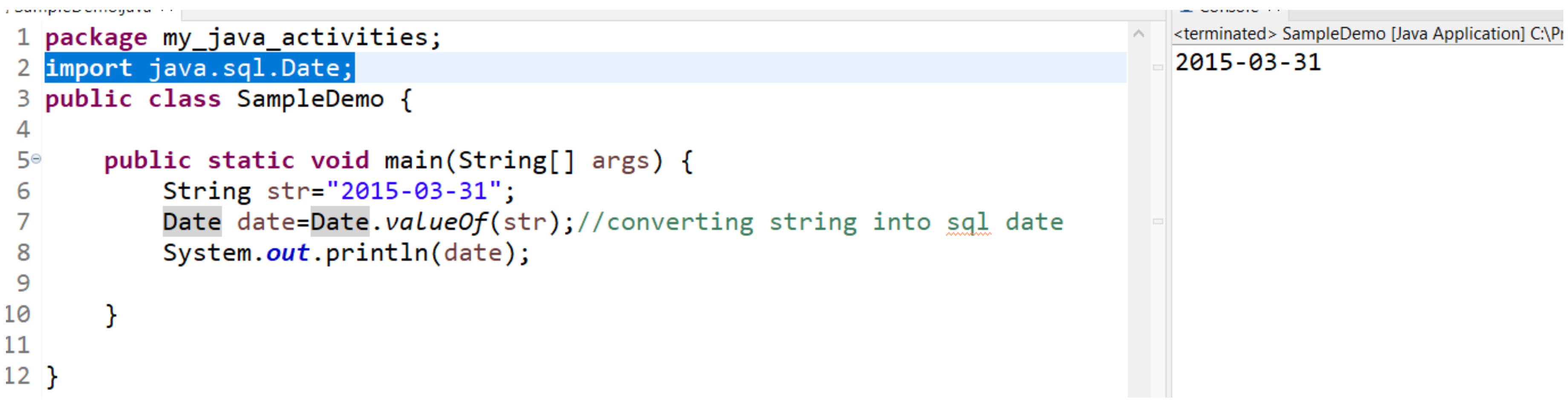


```java
package my_java_activities;
import java.util.ArrayList;
public class SampleDemo {

    public static void main(String[] args) {
          // create an empty array list
          ArrayList<Integer> arrayList = new ArrayList<>();

          // use add() method to add elements in the arrayList
          arrayList.add(20);
          arrayList.add(30);
          arrayList.add(20);
          arrayList.add(30);
          arrayList.add(15);
          arrayList.add(22);
          arrayList.add(11);

          // let us print all the elements available in arrayList
          for (Integer number : arrayList) {
             System.out.println("Number = " + number);
          }

    }

}
```

<terminated> SampleDemo [Java Applic

```
Number = 20
Number = 30
Number = 20
Number = 30
Number = 15
Number = 22
Number = 11
```

Example:Java.sql



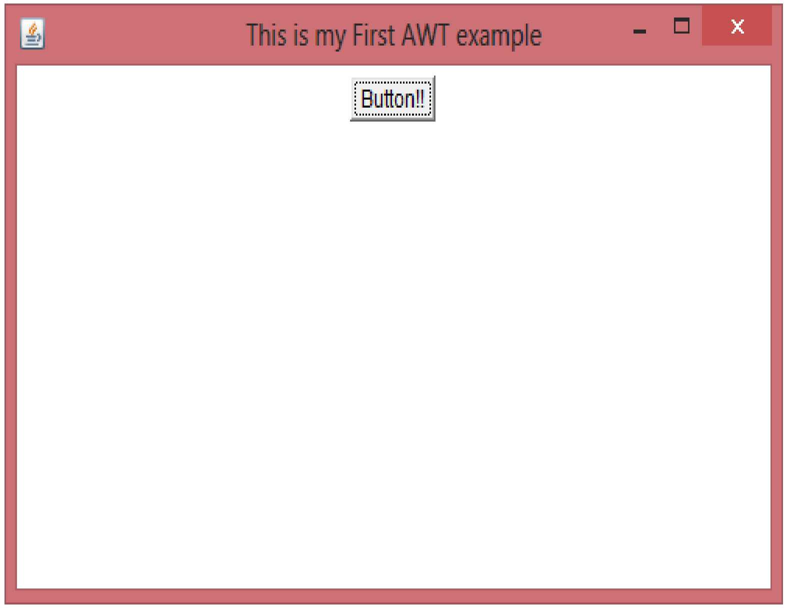
```java
package my_java_activities;
import java.sql.Date;
public class SampleDemo {

    public static void main(String[] args) {
        String str="2015-03-31";
        Date date=Date.valueOf(str);//converting string into sql date
        System.out.println(date);

    }

}
```

```
<terminated> SampleDemo [Java Application] C:\Pr
2015-03-31
```

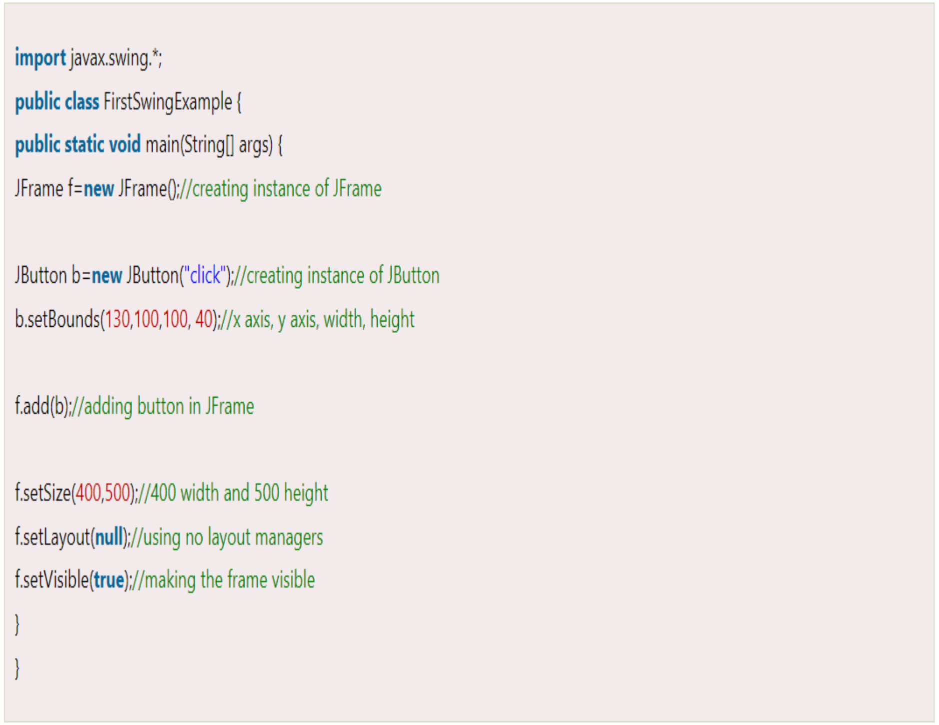
Java.awt

```
import java.awt.*;
// Extended Frame class
// Theclass "SimpleExample" will behave
// like a Frame


public class SimpleExample extends Frame{
   SimpleExample(){
      Button b=new Button("Button!!");

      // setting button position on screen
      b.setBounds(50,50,50,50);

      //adding button into frame
      add(b);

      //Setting Frame width and height
      setSize(500,300);

      //Setting the title of Frame
      setTitle("This is my First AWT example");

      //Setting the layout for the Frame
      setLayout(new FlowLayout());

      /* By default frame is not visible so
       * We are setting the visibility to true
       * To make it visible.
       */
      setVisible(true);
   }
   public static void main(String args[]){
       // Creating the instance of Frame
       SimpleExample fr=new SimpleExample();
   }
}
```

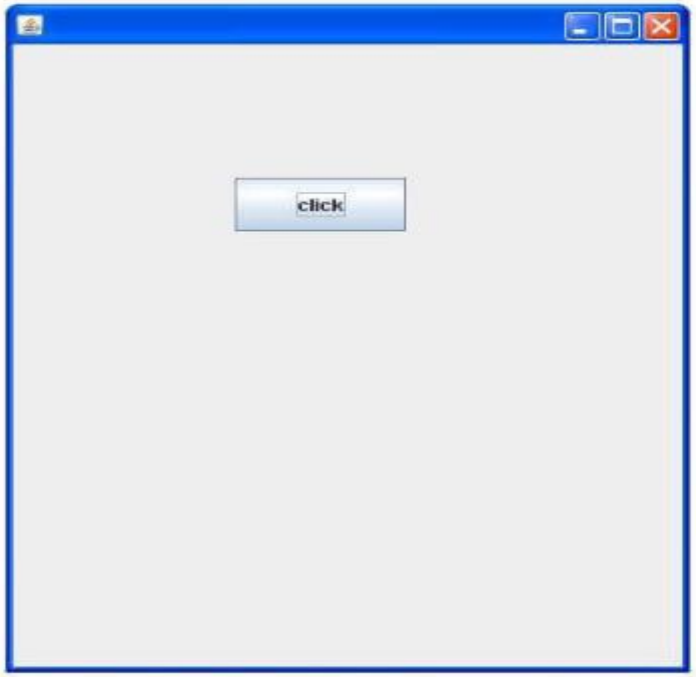
Output



Java.swing



```java
import javax.swing.*;
public class FirstSwingExample {
public static void main(String[] args) {
JFrame f=new JFrame();//creating instance of JFrame

JButton b=new JButton("click");//creating instance of JButton
b.setBounds(130,100,100, 40);//x axis, y axis, width, height

f.add(b);//adding button in JFrame

f.setSize(400,500);//400 width and 500 height
f.setLayout(null);//using no layout managers
f.setVisible(true);//making the frame visible
}
}
```



Java Modifiers

Types of Modifiers

- Access Modifier
- Non –Access Modifier

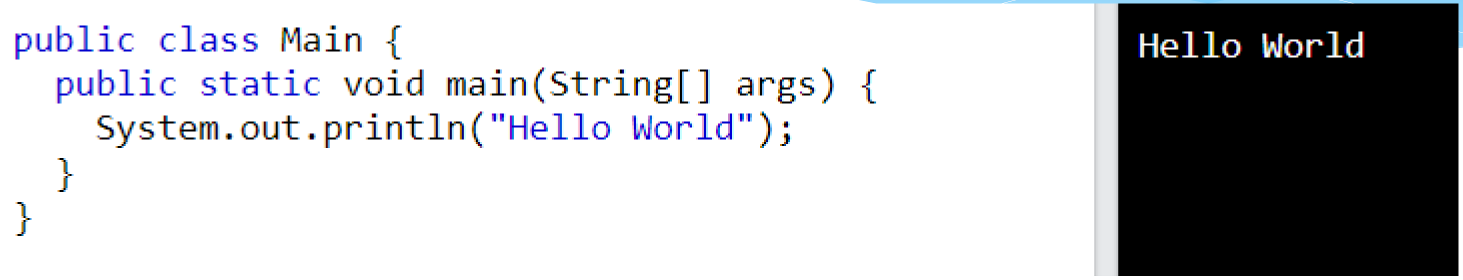
Access Modifier

- For **classes**, you can use either public or *default*

public	The class is accessible by any other class

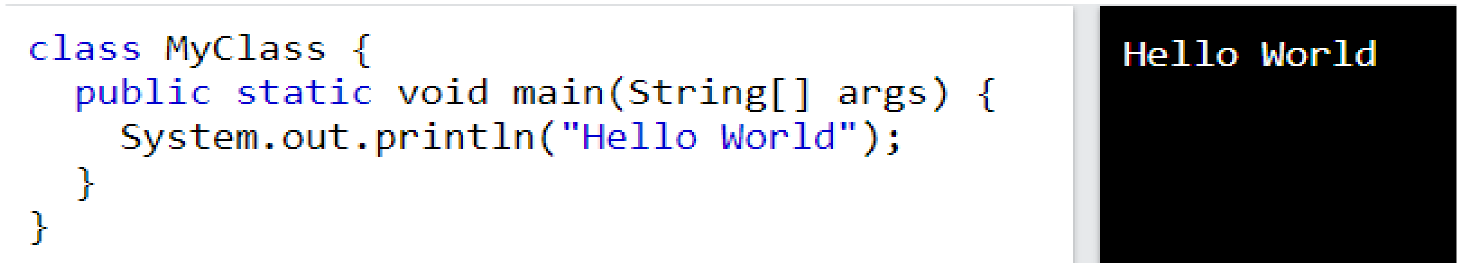
default	The class is only accessible by classes in the same package. This is used when you don't specify a modifier.

Example



```
public class Main {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

```
Hello World
```



```
class MyClass {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

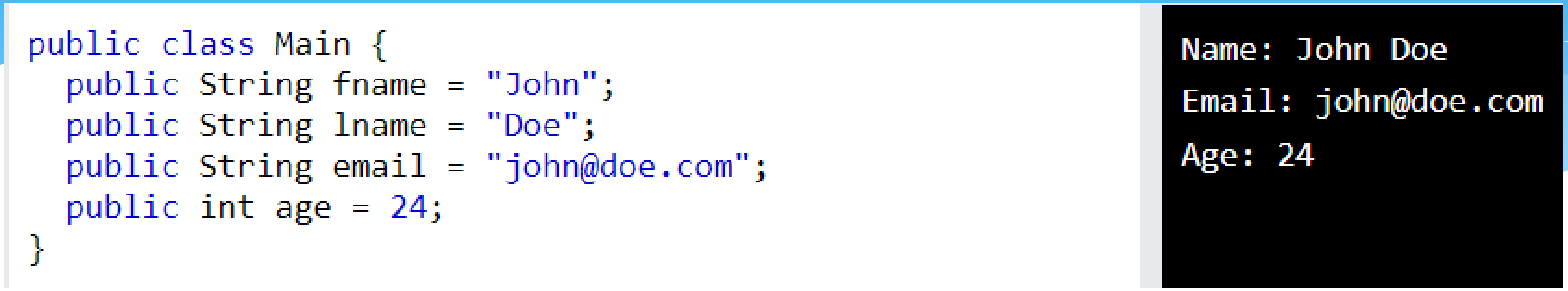
```
Hello World
```

Access Modifier

- For **attributes, methods and constructors**, you can use the one of the following:

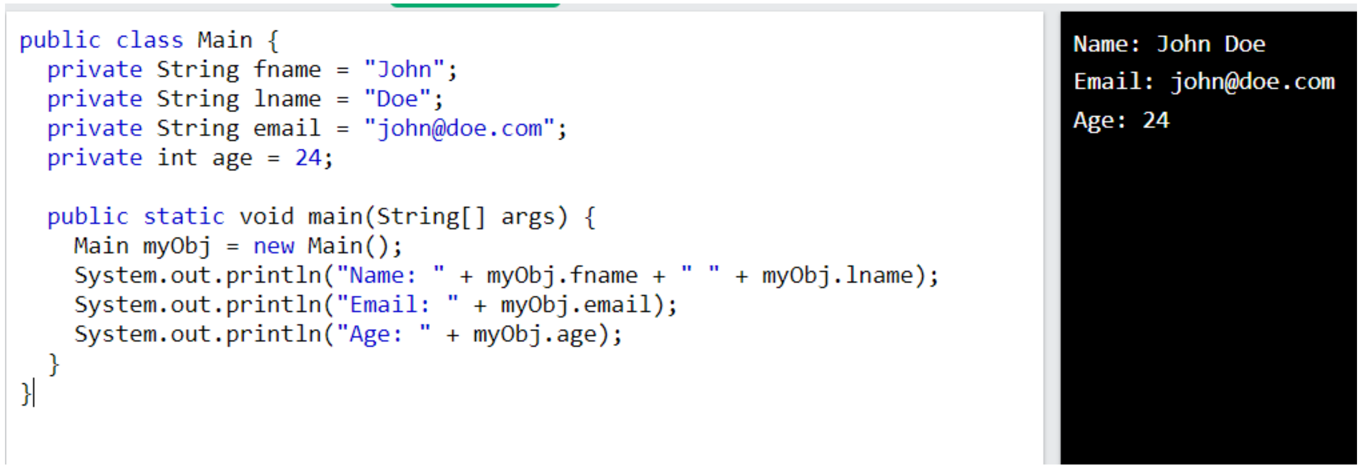
public	The code is accessible for all classes	
private	The code is only accessible within the declared class	
default	The code is only accessible in the same package. This is used when you don't specify a modifier.	
protected	The code is accessible in the same package and **subclasses**.	

Example 1



```
public class Main {
  public String fname = "John";
  public String lname = "Doe";
  public String email = "john@doe.com";
  public int age = 24;
}
```

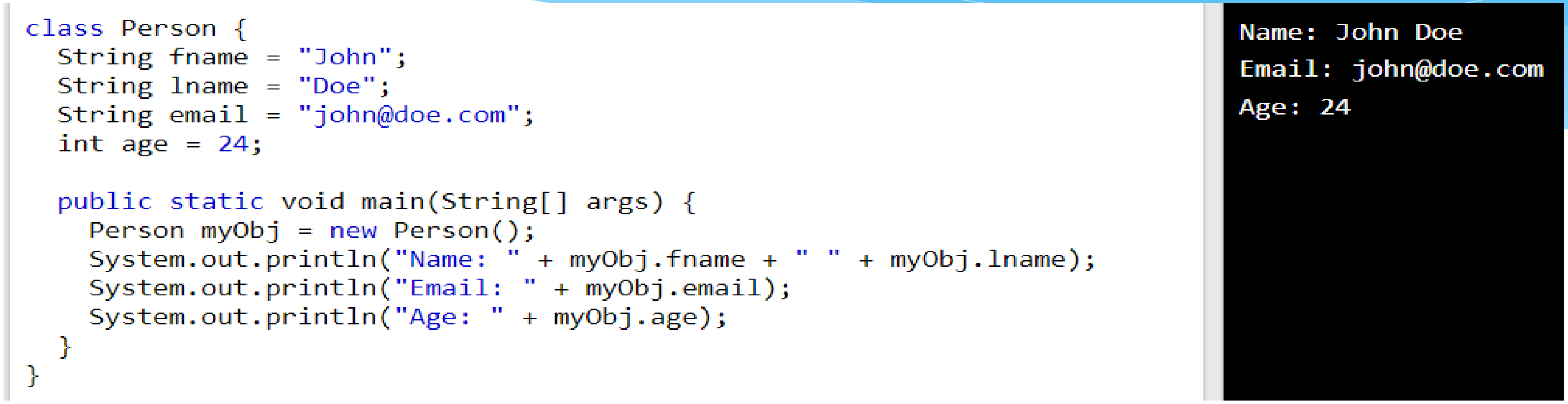
```
Name: John Doe
Email: john@doe.com
Age: 24
```



```
public class Main {
  private String fname = "John";
  private String lname = "Doe";
  private String email = "john@doe.com";
  private int age = 24;

  public static void main(String[] args) {
    Main myObj = new Main();
    System.out.println("Name: " + myObj.fname + " " + myObj.lname);
    System.out.println("Email: " + myObj.email);
    System.out.println("Age: " + myObj.age);
  }
}
```

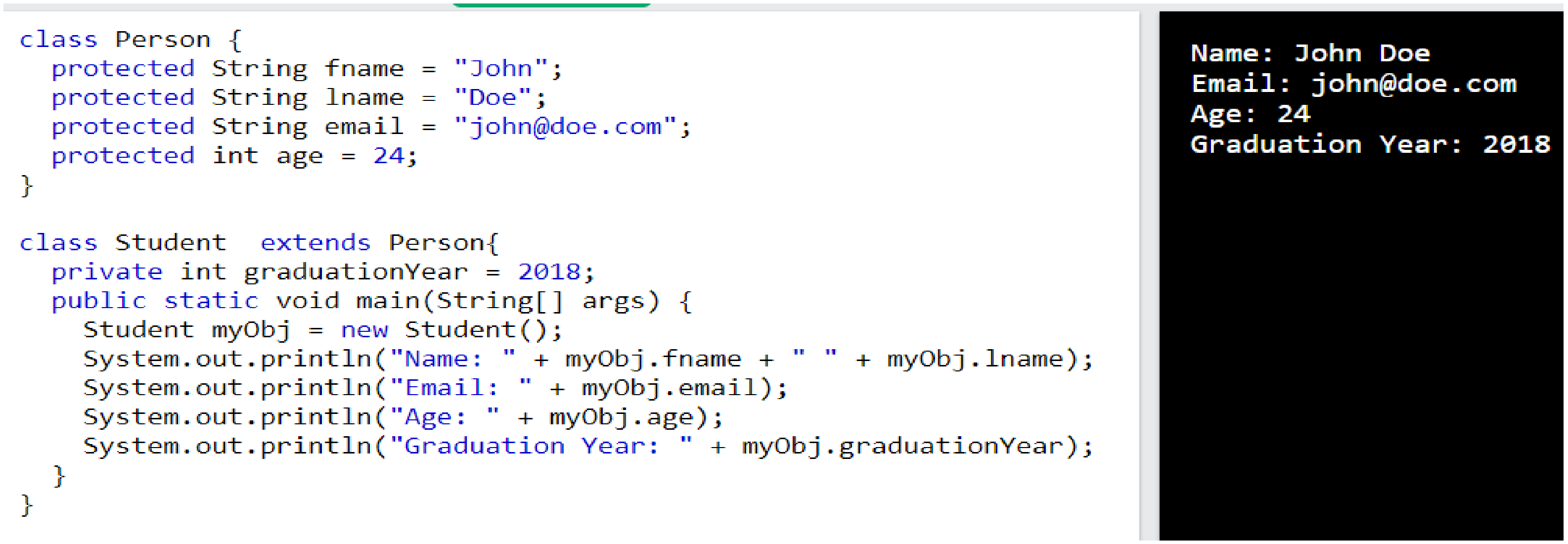
```
Name: John Doe
Email: john@doe.com
Age: 24
```

Example2



```java
class Person {
  String fname = "John";
  String lname = "Doe";
  String email = "john@doe.com";
  int age = 24;

  public static void main(String[] args) {
    Person myObj = new Person();
    System.out.println("Name: " + myObj.fname + " " + myObj.lname);
    System.out.println("Email: " + myObj.email);
    System.out.println("Age: " + myObj.age);
  }
}
```

```
Name: John Doe
Email: john@doe.com
Age: 24
```



```java
class Person {
  protected String fname = "John";
  protected String lname = "Doe";
  protected String email = "john@doe.com";
  protected int age = 24;
}

class Student  extends Person{
  private int graduationYear = 2018;
  public static void main(String[] args) {
    Student myObj = new Student();
    System.out.println("Name: " + myObj.fname + " " + myObj.lname);
    System.out.println("Email: " + myObj.email);
    System.out.println("Age: " + myObj.age);
    System.out.println("Graduation Year: " + myObj.graduationYear);
  }
}
```

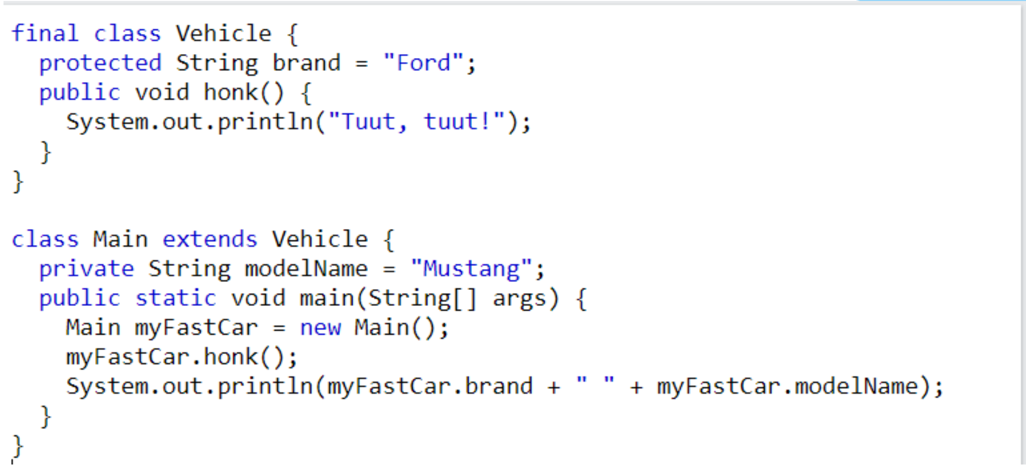
```
Name: John Doe
Email: john@doe.com
Age: 24
Graduation Year: 2018
```

Non-Access Modifiers

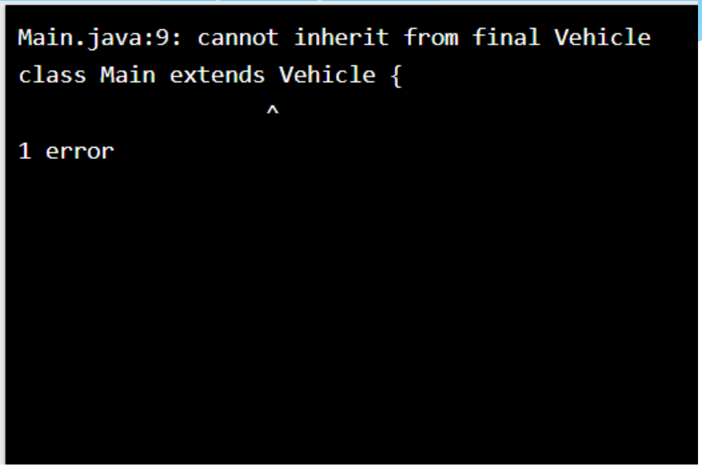
For **classes**, you can use either final or abstract:

final	The class cannot be inherited by other classes	
abstract	The class cannot be used to create objects (To access an abstract class, it must be inherited from another class. You will learn more about inheritance and abstraction in the Inheritance and Abstraction	

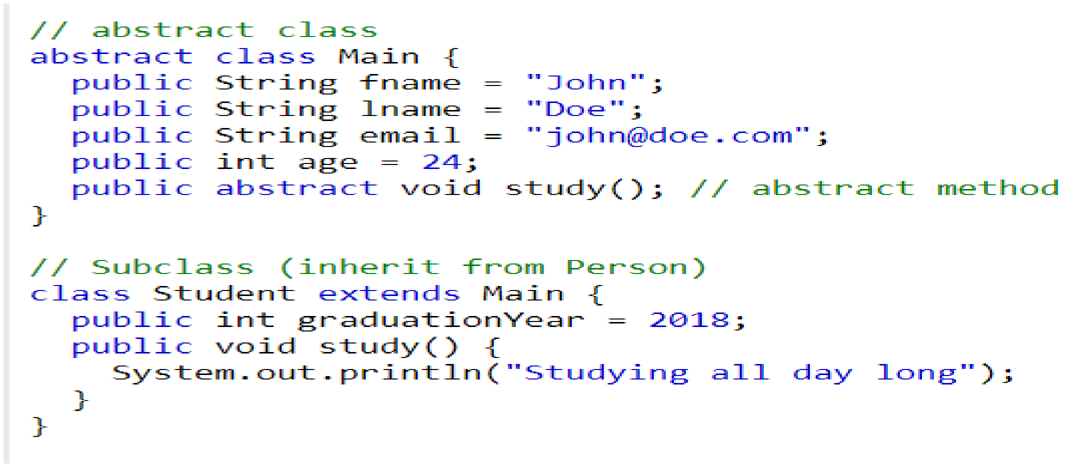
Example



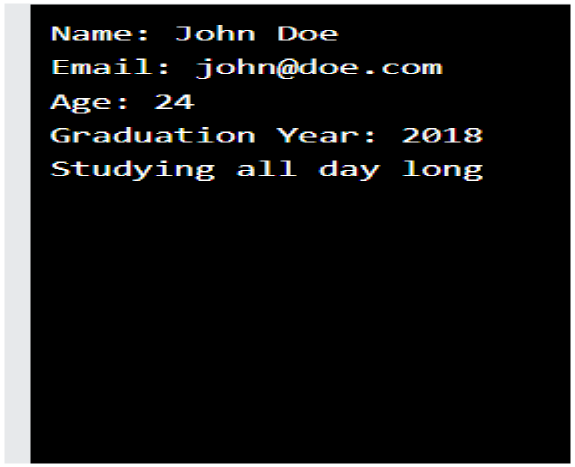
```java
final class Vehicle {
  protected String brand = "Ford";
  public void honk() {
    System.out.println("Tuut, tuut!");
  }
}

class Main extends Vehicle {
  private String modelName = "Mustang";
  public static void main(String[] args) {
    Main myFastCar = new Main();
    myFastCar.honk();
    System.out.println(myFastCar.brand + " " + myFastCar.modelName);
  }
}
```



```
Main.java:9: cannot inherit from final Vehicle
class Main extends Vehicle {
                  ^
1 error
```



```java
// abstract class
abstract class Main {
  public String fname = "John";
  public String lname = "Doe";
  public String email = "john@doe.com";
  public int age = 24;
  public abstract void study(); // abstract method
}

// Subclass (inherit from Person)
class Student extends Main {
  public int graduationYear = 2018;
  public void study() {
    System.out.println("Studying all day long");
  }
}
```



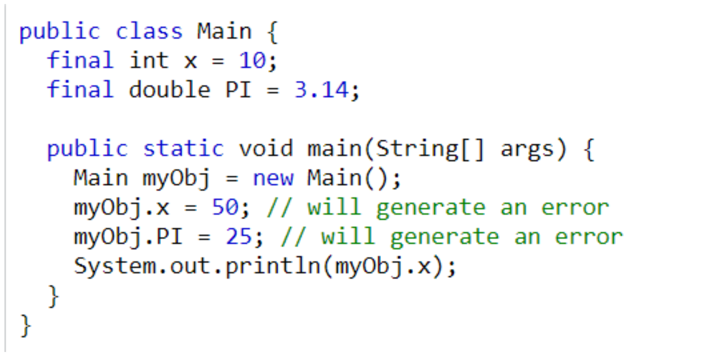
```
Name: John Doe
Email: john@doe.com
Age: 24
Graduation Year: 2018
Studying all day long
```

Non-Access Modifiers

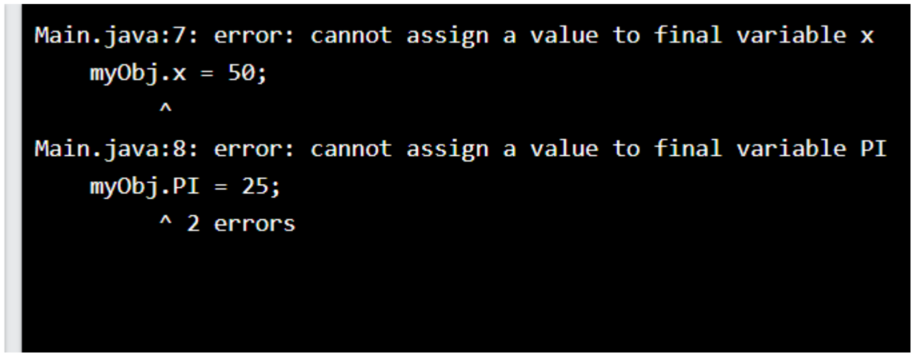
For **attributes and methods**, you can use the one of the following:

final	Attributes and methods cannot be overridden/modified
static	Attributes and methods belongs to the class, rather than an object

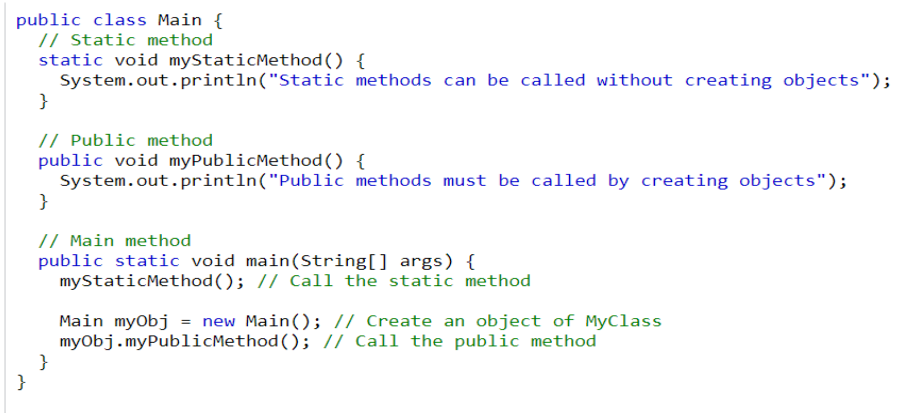
Example



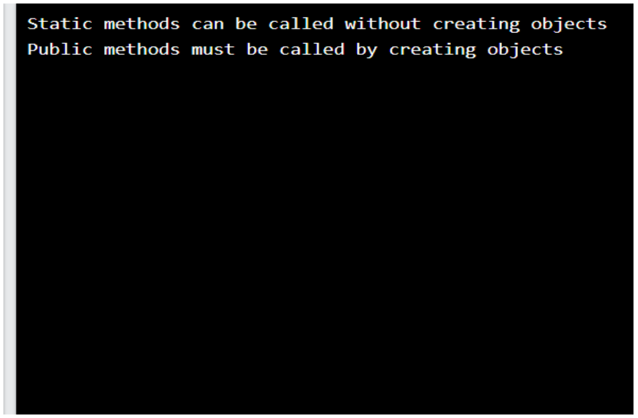
```java
public class Main {
  final int x = 10;
  final double PI = 3.14;

  public static void main(String[] args) {
    Main myObj = new Main();
    myObj.x = 50; // will generate an error
    myObj.PI = 25; // will generate an error
    System.out.println(myObj.x);
  }
}
```



```
Main.java:7: error: cannot assign a value to final variable x
    myObj.x = 50;
         ^
Main.java:8: error: cannot assign a value to final variable PI
    myObj.PI = 25;
         ^ 2 errors
```



```java
public class Main {
  // Static method
  static void myStaticMethod() {
    System.out.println("Static methods can be called without creating objects");
  }

  // Public method
  public void myPublicMethod() {
    System.out.println("Public methods must be called by creating objects");
  }

  // Main method
  public static void main(String[] args) {
    myStaticMethod(); // Call the static method

    Main myObj = new Main(); // Create an object of MyClass
    myObj.myPublicMethod(); // Call the public method
  }
}
```



```
Static methods can be called without creating objects
Public methods must be called by creating objects
```