Parul® University

Object Oriented Programming with JAVA

Ms.Vaishalee Joishar,Cyber Secutiy Trainer

Parul®
University

DIGITAL LEARNING CONTENT

UNIT-7

String , Packages and InterFaces

STRING

- A **String** is a collection of characters. In Java, a string is an object that represents a collection of objects. A string is a predefined class used to create string objects. It is an **immutable object**, which means it can't be updated once created.
- How to create a string object?
  - There are two ways to create String object:

    - **By string literal:-** Java String literal is created by using double quotes. For Example: String name = "Parul University";

    - **By new keyword:-**String str=new String("Welcome");//creates two objects and one reference variable

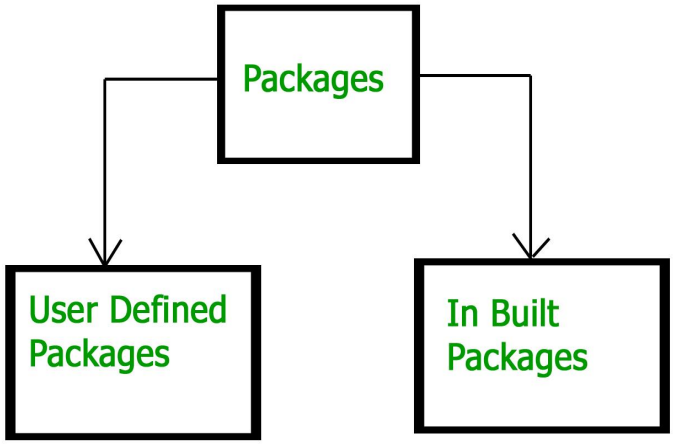
Cont...

The string class has a set of built-in-methods, Some methods defined below:-

- charAt(): It returns a character at a specified position.
- equals(): It compares the two given strings and returns a Boolean, that is, True or False.
- concat(): Appends one string to the end of another.
- length(): Returns the length of a specified string.
- toLowerCase(): Converts the string to lowercase letters.
- toUpperCase(): Converts the string to uppercase letters.
- indexOf(): Returns the first found position of a character.
- substring(): Extracts the substring based on index values, passed as an argument.

PACKAGES

- Packages in Java serve as a container for organizing classes, interfaces, and sub-packages with similar functionalities.
- Package in java can be categorized in two form, built-in package and user-defined package.



- **Built-in packages:** Built-in packages is also known as 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.

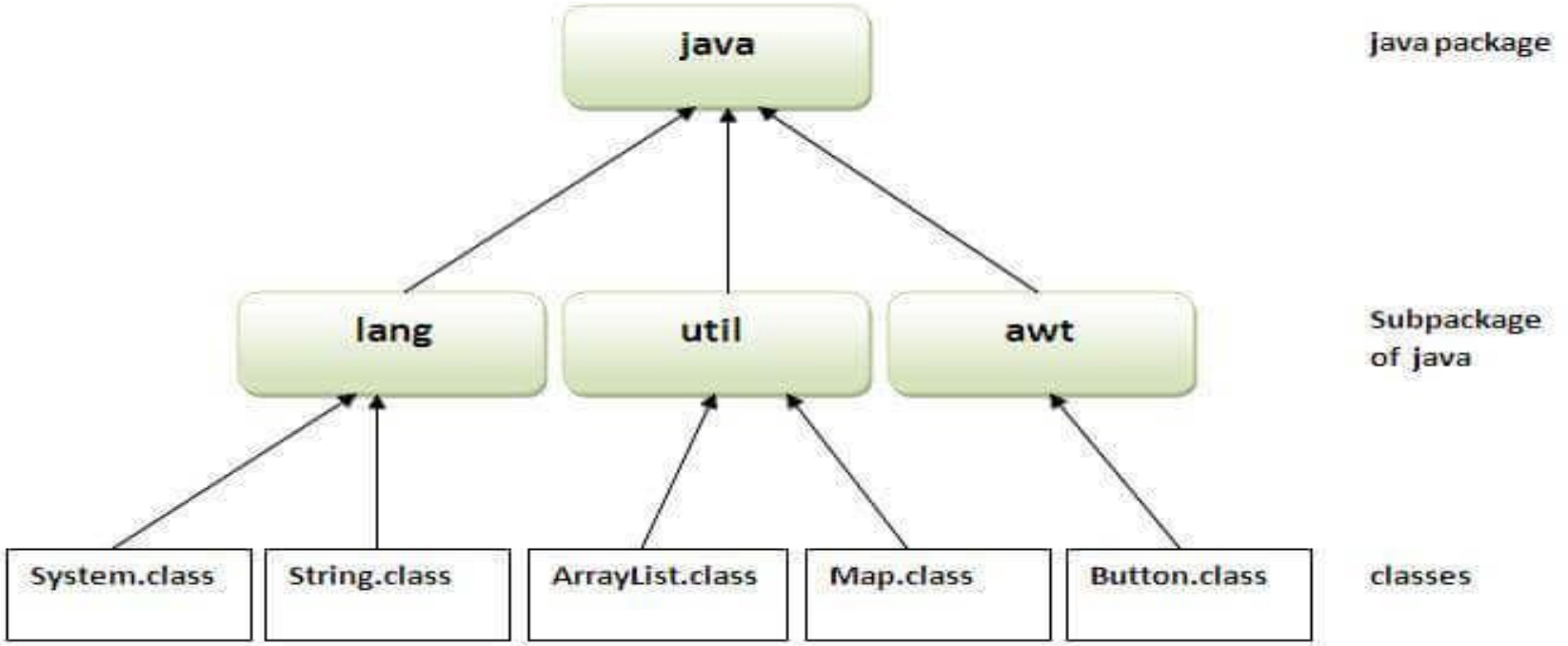
Advantages of Packages

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.

built-in package



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.

Parul® University

Example of java package

Build-in package

```
import java.lang.*;

class example_lang {
  public static void main(String args []) {
     int a = 100, b = 200,maxiNumber;
     maxiNumber = Math.max(a,b);
     System.out.printf("Maximum Number is = "+maxiNumber);
   }
}
```

How to run java package program:

- To Compile: javac example_lang.java
- To Run: java example_lang

User Defined Package

```
package UserPackage;

public class UserClass {
    Run | Debug
    public static void main(String args[]) {
        System.out.println(x:"UserPAckege creates a UserClass !!!");
    }
}
```

How to run java user defined package program:

- To Compile: javac -d . UserClass.java
- To Run: java Userpackage. UserClass

How to access package from another package?

1. import package.*;
2. import package.classname;
3. fully qualified name.

1) Using packagename.*

- If you use package.* then all the classes and interfaces of this package will be accessible but not subpackages.
- The import keyword is used to make the classes and interface of another package accessible to the current package.

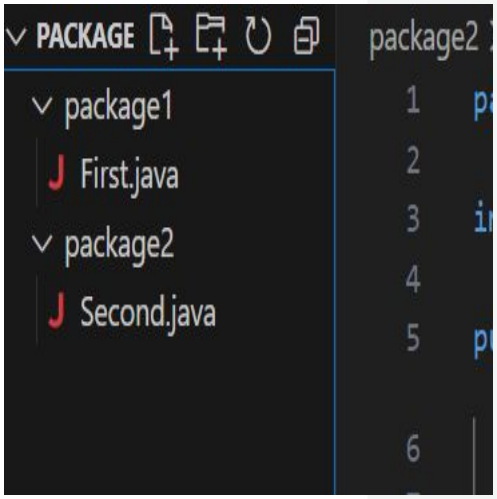
2) import package.classname;

- If you import package.classname then only declared class of this package will be accessible.

3) Using fully qualified name.

- If you use fully qualified name then only declared class of this package will be accessible. Now there is no need to import.

1) import the packagename.*




```
package package1;

public class First {
    public void msg() {
        System.out.println(x:"Hello");
    }
}
```

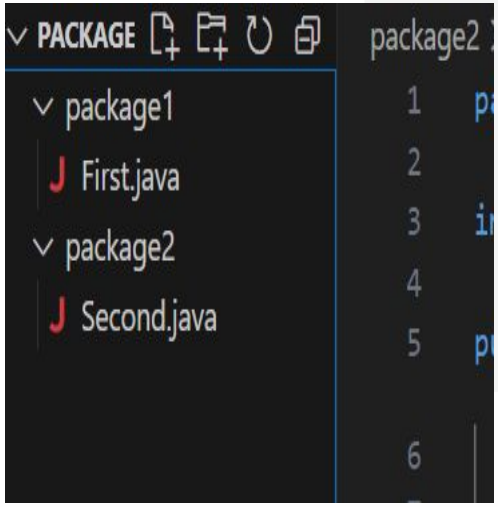
```
package package2;

import package1.*;

public class Second {
    Run | Debug
    public static void main(String args[]) {
        First obj = new First();
        obj.msg();
    }
}
```

2) Using packagename.classname

- If you import package.classname then only declared class of this package will be accessible.




```java
package package1;

public class First {
    public void msg() {
        System.out.println(x:"Hello");
    }
}
```

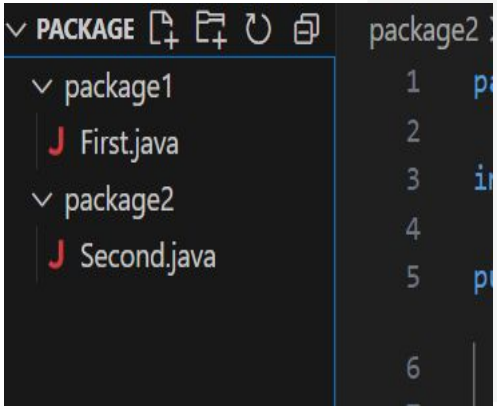
```java
package package2;

import package1.First;

public class Second {
    Run | Debug
    public static void main(String args[]) {
        First obj = new First();
        obj.msg();
    }
}
```

3) Using fully qualified name

- If you use fully qualified name then only declared class of this package will be accessible. Now there is no need to import.




```
package package1;

public class First {
  public void msg() {
    System.out.println(x:"Hello");
  }
}
```

```
package package2;

import package1.First;

public class Second {
  Run | Debug
  public static void main(String args[]) {
    package1.First obj = new package1.First();
    obj.msg();
  }
}
```

Classpath

There are two ways to load the class files temporary and permanent.

- Temporary
  - By setting the classpath in the command prompt
  - By -classpath switch

- Permanent
  - By setting the classpath in the environment variables
  - By creating the jar file, that contains all the class files, and copying the jar file in the jre/lib/ext folder.

Interface

- An **interface in Java** is a blueprint of a class. It has static constants and abstract methods.
- The interface in Java is *a mechanism to achieve* **abstraction.** There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and **multiple inheritance in java.**
- In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.
- Java Interface also **represents the IS-A relationship**.
- It cannot be instantiated just like the abstract class.
- Since Java 8, we can have **default and static methods** in an interface.
- Since Java 9, we can have **private methods** in an interface.

Why use Java interface and how to Declare

There are mainly three reasons to use interface.

Declare an interface

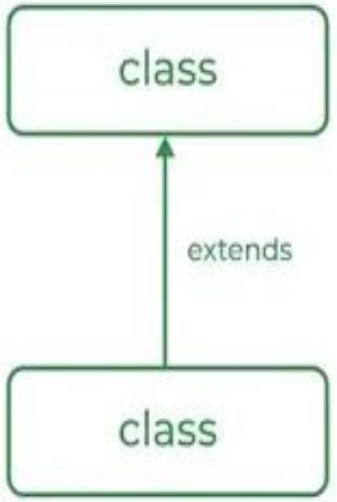
- An interface is declared by using the interface keyword. It provides total abstraction; means all the methods in an interface are declared with the empty body, and all the fields are public, static and final by default. A class that implements an interface must implement all the methods declared in the interface.

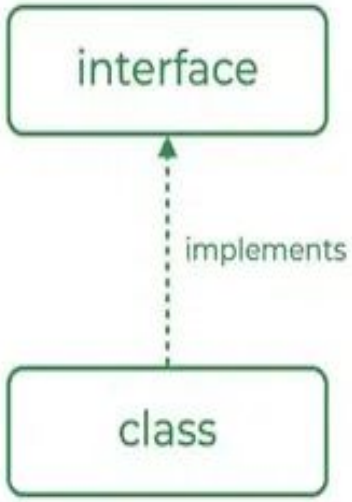
Syntax for Java Interfaces

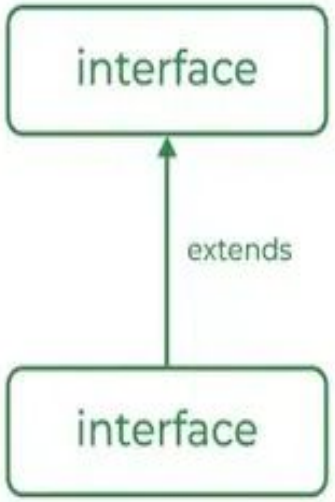
- interface interfaceName{
  // declare constant fields
  // declare methods that abstract
  // by default.
  }

Relationship Between Class and Interface

- A class can extend another class similar to this an interface can extend another interface. But only a class can extend to another interface.

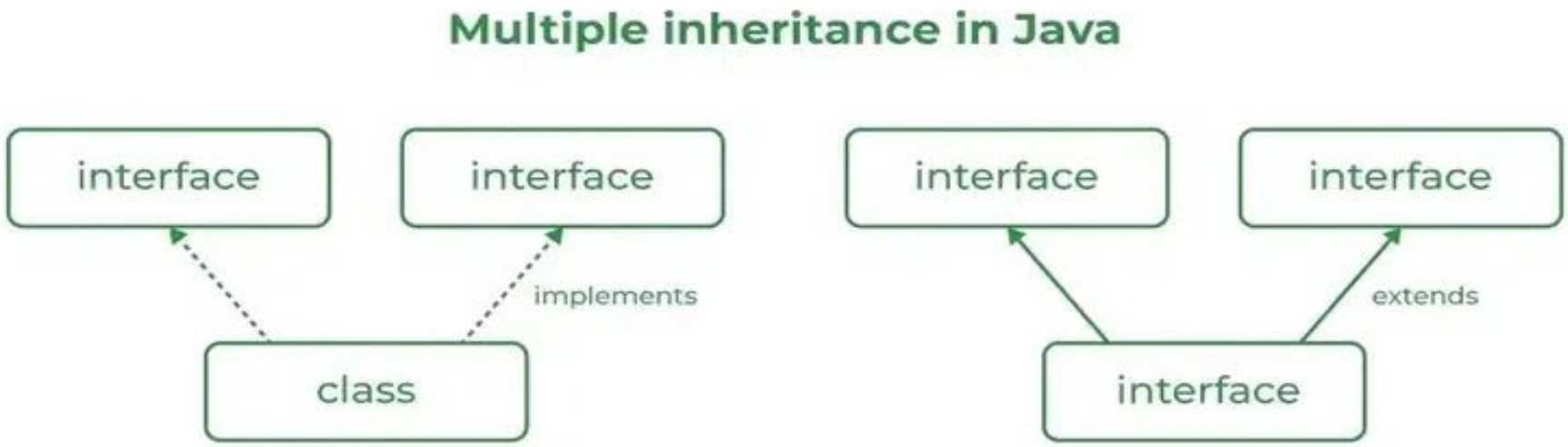






Multiple Inheritance in Java Using Interface

- Multiple Inheritance is an OOPs concept that can't be implemented in Java using classes. But we can use multiple inheritances in Java using Interface.
- ***Implementation:*** *To implement an interface, we use the keyword* ***implements***



Difference Between Class and Interface

Class	Interface
In class, you can instantiate variables and create an object.	In an interface, you can't instantiate variables and create an object.
A class can contain concrete (with implementation) methods	The interface cannot contain concrete (with implementation) methods.
The access specifiers used with classes are private, protected, and public.	In Interface only one specifier is used- Public.

Enum in Java

- **Enumeration (Enum)** in Java was introduced in JDK 1.5. Most of the other programming languages like C, C++, etc also provide support for enumerations.
- **Enum** is a keyword, a feature which is used to represent a fixed number of well-known values in Java, For example, number of days in a week.
- **Enum** constants are implicitly static and final and you can not change their value once created. In short, enumeration is a list of named constants.
- **Enum** in Java provides type-safety and can be used inside switch statements like int variables.
- A Java enumeration is a class type. Although we don’t need to instantiate an enum using **new,** it has the same capabilities as other classes. Just like classes, you can give them constructors, add instance variables and methods, and even implement interfaces.

Declaration of Enum

Enum declaration can be done outside a Class or inside a Class but not inside a Method.

1. Declaration outside the class

```
enum Color {
    RED,
    GREEN,
    BLUE;
}

public class EnumerationClass {
    // Driver method
    public static void main(String[] args)
    {
        Color c1 = Color.RED;
        System.out.println(c1);
    }
}
```

2. Declaration inside a class

```
public class EnumerationClass {
    enum Color {
        RED,
        GREEN,
        BLUE;
    }

    public static void main(String[] args)
    {
        Color c1 = Color.RED;
        System.out.println(c1);
    }
}
```

Properties of Enum

There are certain properties followed by Enum as mentioned below:

- Every enum is internally implemented by using Class.
- Every enum constant represents an **object** of type enum.
- Enum type can be passed as an argument to **switch** statements.
- Every enum constant is always implicitly **public static final**. Since it is **static**, we can access it by using the enum Name. Since it is **final**, we can't create child enums.
- We can declare the **main() method** inside the enum. Hence we can invoke the enum directly from the Command Prompt.

Java Enum Programs

1. Main Function Inside Enum

We can declare a main function inside an enum as we can invoke the enum directly from the Command Prompt.

```
enum Color {
    red,
    purple,
    yellow;

    public static void main(String[] args)
    {
        Color c = Color.yellow;
        System.out.println(c);
    }
}
```

Cont...

2. Loop through Enum

- We can iterate over the Enum using values( ) and loop. values() function returns an array of Enum values as constants using which we can iterate over the values.

```
enum Color {
      red,
    purple,
    yellow;
}
class LoopEnum{

    public static void main(String[] args)
    {
        for (Color var_1 : Color.values()) {
            System.out.println(var_1);
        }
    }
}
```

Cont...

3. Enum in a Switch Statement

```
class SwitchEnum {
    enum Color {
      red,
    purple,
    cyan,
    yellow;
}
    public static void main(String[] args)
    {
          Color var_1=Color.yellow;
         switch(var_1){
          case red:
            System.out.println("Red color observed");
            break;
          case purple:
            System.out.println("Green color observed");
            break;
          case cyan:
            System.out.println("Blue color observed");
          default:
            System.out.println("Other color observed");
        }
    }
}
```

There are mainly 3 Methods used in Enum

1.values( ) method

In Java, the values( ) method can be used to return all values present inside the enum.

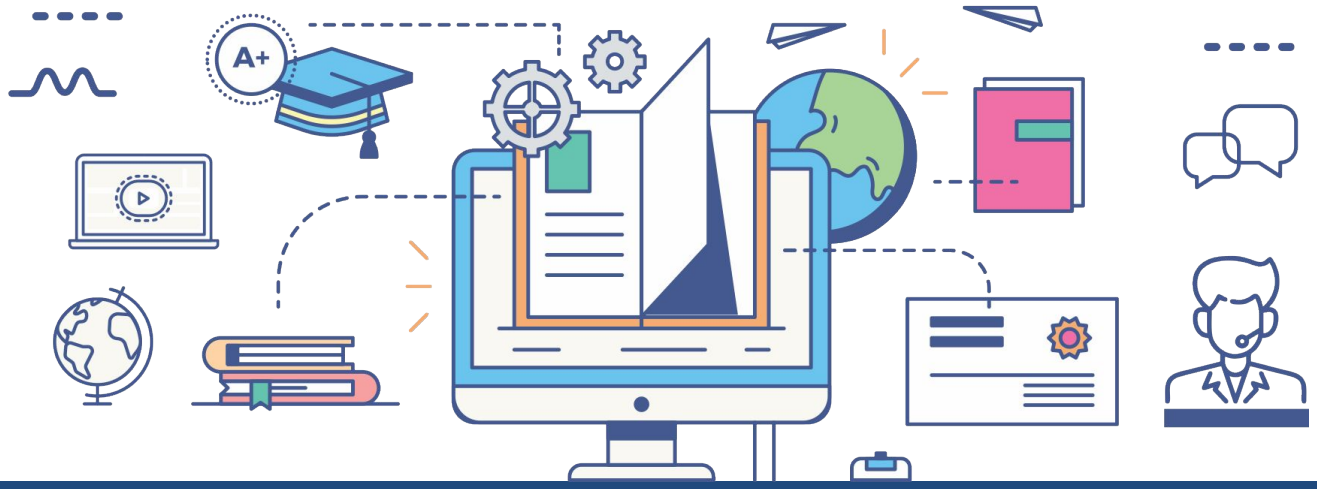
2. valueOf() method

The valueOf() method returns the enum constant of the specified string value if exists.

3. ordinal() method

By using the ordinal() method, each enum constant index can be found, just like an array index.

DIGITAL LEARNING CONTENT



Parul® University

www.paruluniversity.ac.in