Agenda

1. Interfaces & Abstract Class
2. Wrapper Classes
3. String, StringBuffer & StringBuilder
4. Garbage Collection, Reflection

Interface, Why Interface, What is the difference between interface & abstract class

Interface in JAVA is used to provide specification.

Real- World Example : Surge protector – It can be connected with any electrical & electronic devices

* Smart TV – draw different power (star ratings, Total wattage it consumes different amount of power)
* Washing Machine
* Refrigerator
* Mobile phone chargers
* TubeLight ……

By default, unless specified the methods defined inside interface is always public and abstract.

Eclipse is a costlier process – [If a program/application consumes more memory or more processing power then it is called costlier process]

Interface can have non-abstract methods.

Abstract methods will not have implementation/ body – Abstract methods are non-empty methods

<https://www.javatpoint.com/difference-between-abstract-class-and-interface>

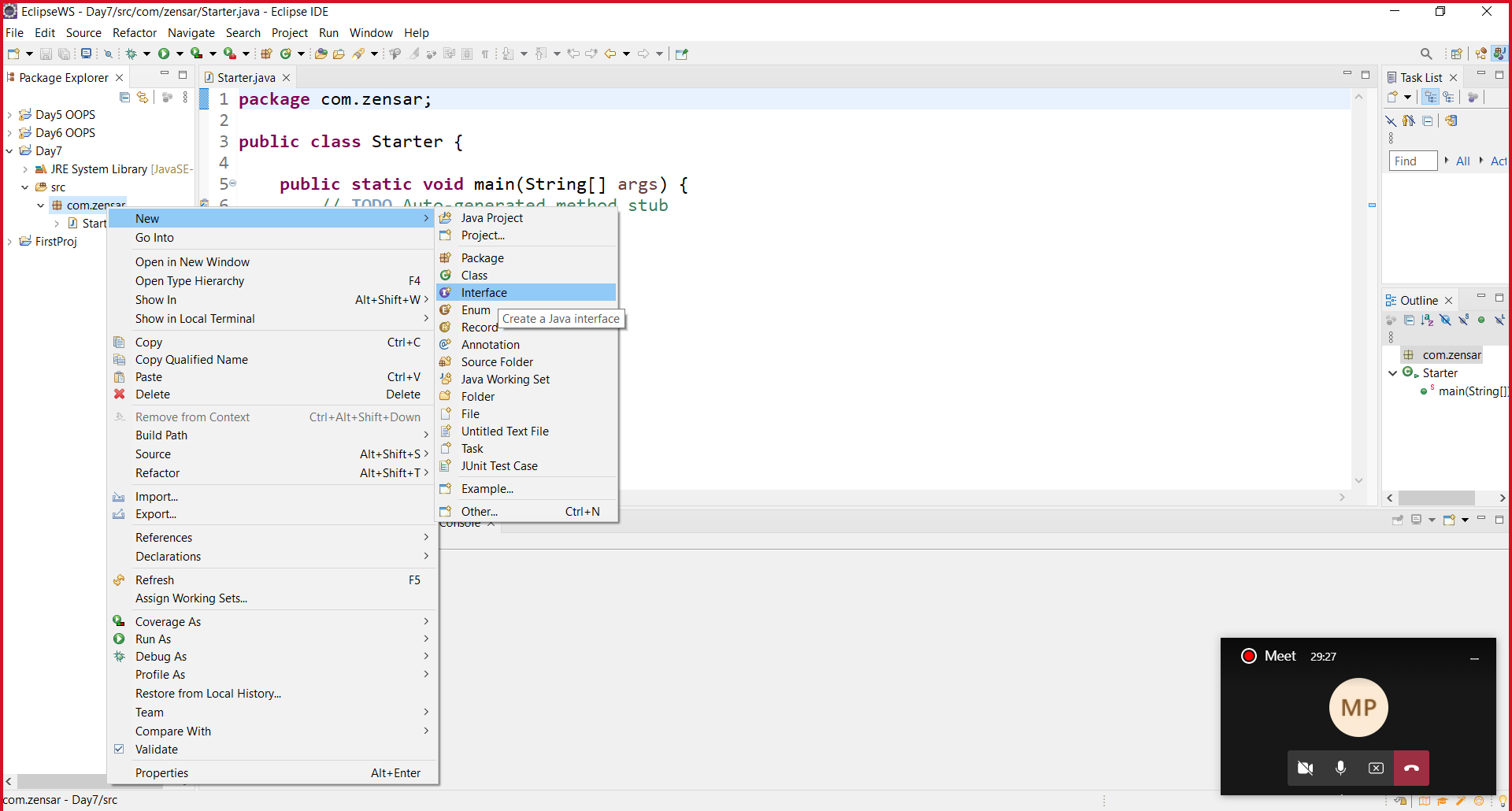
<https://www.geeksforgeeks.org/difference-between-abstract-class-and-interface-in-java/>

To create a new interface

1. Open Eclipse EE version then select new menu & select an interface

Empty methods & abstract methods are not equal

Empty methods will have body but abstract methods won’t have body.



String Class

In Java, String is a special Class. Bcos this is the only class which supports object method of declaration and primitive method of declaration.

String by default immutable. It value can’t be modified after creation.

When we try to modify a String object/literal it will create another copy in the String constant pool.

String class is defined in java.lang package.

When we use extends keyword, we should not use a final class

Extends keyword can’t be followed by final classes (Final classes can’t be extended)

Strings are constant by default. Their values can’t be modified after creation.

UTF – Unicode Text Format (16 bits)

In Java, char – 2 bytes, 16 bits.

Quiz –

1. Assume I’m giving you 1000 one-rupee coin & 10 boxes
2. I’m asking you to put these 1000 one-rupee coins in these 10 boxes in any order.
3. After putting all the coins in boxes, I will ask you any amount between 1 – 1000
4. The least value I can ask from you is One rupee
5. The max value I could ask is 1000 rupee
6. I assure that I’ll not you any amount in fraction (500, 5001 not ~~500.50, 585.75)~~
7. While giving the money, make sure to give the exact amount what I asked for
8. While giving me the money you can give one or more boxes without opening it.
9. When a particular value is asked, you can’t open the box and you should not take any amount partially from the box.
10. What will be the order – that you use to put coins in 10 boxes.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 489 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

1000-511 = 489

16 bits = 2^15 = Max value

8 bits = 2 ^7 = 128 (-127 to 128, 0-255)

In C and other programming langs, char datatype size is 1 byte only (8 bits) (0-255)

A-Z = 26

a-z = 26

0-9 = 10

Hindi, Telugu, Kannada, Tamil (247 characters) 255- 247 =

When we use two index (Start & End) with java String functions, “Start index in inclusive & End-index is exclusive”

This example is related to Binary number system.

StringBuffer (ThreadSafe Mutable charSequence) & StringBuilder (Non-ThreadSafe Mutable Char-Sequence)

Package – package is a keyword in java which is used to group similar set of classes & interfaces

There are two types of packages

1. Built-in /System defined package (java.lang, java.util, java.io, java.sql – Language creators developed package)
2. User defined package/Custom package (usually reverse of the company url, com.zensar.projectname.modulename)

Defining the name of the package should be the first line in your java source code.

Package will help you to resolve naming conflict.

Package name should be in lower case only.

In a project, we normally group classes/interfaces of similar nature (controller, model/entity, repository/dao, service, util)

Com.zensar.mybank – base package name

Com.zensar.mybank.user

Com.zensar.mybank.auth

Com.zensar.mybank.controller

Com.zensar.mybank.service

Com.zensar.mybank.repository

Con.zensar.mybank.dao

Wrapper Classes – It’s a wrapper to convert primitive data type to it’s corresponding object.

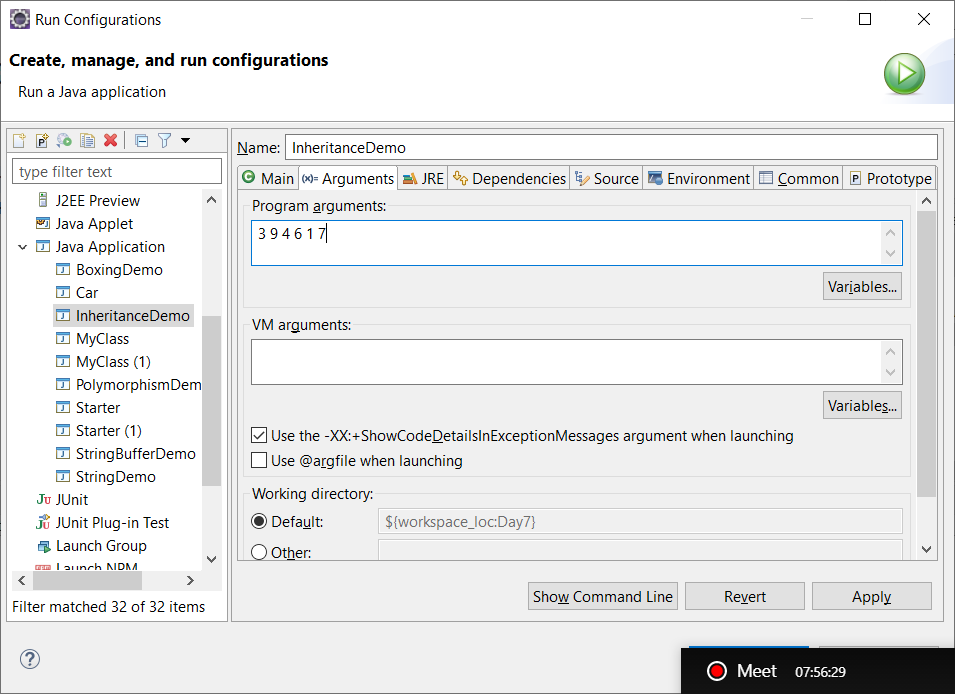
|  |  |  |
| --- | --- | --- |
|  | Primitive | Wrapper |
| 1 | boolean | Boolean |
| 2 | byte | Byte |
| 3 | char | Character |
| 4 | int | Integer |
| 5 | float | Float |
| 6 | double | Double |
| 7 | long | Long |
| 8 | short | Short |

Converting primitive to its corresponding object is called Boxing (primitive – object – boxing)

Reverse of Boxing is called unboxing. (object-primitive unboxing)

JVM will do both boxing & unboxing automatically.

Boxing & Unboxing – Collection



**package** com.zensar;

**public** **class** CLArgs {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** firstValue = Integer.*parseInt*(args[0]);

**for** (**int** i=1;i<args.length;i++) {

}

}

}