Java

* Java is platform independent
* Object Oriented Programming

4 Main Features

1. Encapsulation
2. Inheritance
3. Polymorphism
4. Abstraction

Encapsulation: Wrapping data & methods in a single unit, and making members of data private & accessing them through setters & getters.

* You can avoid unrealistic values like age being -ve or greater than 100, seconds/minutes beyond 60
* You can give write access only for certain variables so that the data can’t be modified once its initialized.

Employee emp = new Employee(100, “Alex”);

emp -> different code -> you must not able to modify, like emp id must always be 100, in that case you can avoid providing setter method of id

If setId() is not present, then you can’t access setId() at all

class A -> emp -> class B ->

Create an Address property part of the Person, the Address class must have state, city & pin code.

You must create an encapsulated class.

Inheritance:

You can acquire common properties & behaviour from a class to another class, you can use extends keyword to achieve inheritance.

super():

Polymorphism

Single method with many forms, it can be achieved in 2 ways

1. overloading - same method with different signature like parameters & type of parameters in the same class
2. overriding - same method & same signature like parameters & type of parameters, but with different logics in the subclass & parent class

Ex: Power button in TV/Remote/Laptop, single operation but many forms.

display(): it must have different forms.

Abstraction:

Hiding the complexity and showing the necessary details to the user

interface & abstract class

Interface: You can only create abstract methods & constants, Different classes can implement interfaces in different way.

Some important points about interface

* You can’t have constructors
* You can’t implement methods but from java 8 onwards interfaces can have default & static methods
* You can’t create object of an interface
* You can declare variables of interface type

ex: List items; here List is an interface

Some important points about the abstract class

* You can create constructors
* You can have both abstract & non-abstract methods
* You can’t create object of an abstract class
* You can declare variables of abstract class type

Exceptions:

They are runtime errors which abnormally terminates the program.

You have 5 keywords in exceptions

try, catch, finally, throw & throws.

try: statements that might cause exceptions

catch: handles the exceptions generated in try-block

finally: Definitely executed whether or not exceptions handled

throw: It is to manually create an exception

throws: It is to pass the exception to the caller, so that caller can handle it.

Some Java API’s

Wrapper classes - Integer, Long, Boolean, Float and etc.

String & String Buffer

Object

Scanner

Date

Calendar

Wrapper classes: These are the classes provided for every primitive datatypes, one thing we can do in primitive is only arithmetic related operations, but with wrapper classes we can do extra operations like converting from one type to another, checking the max/min limit of the type

Byte.MIN\_VALUE >> -128

Byte.MAX\_VALUE >> 127

Integer.MAX\_Value & Integer.MIN\_VALUE provides min and max value of int

You can convert string to int, double, boolean using some static method.

Integer.parseInt(“123”); 123 in string is converted to int

Double.parseDouble(“265.35”); 265.35 in string is converted to double.

Comparing two numbers to find the biggest / smallest number using compare method

The compare method returns -1, 0 , 1 when 2 numbers are compared.

-1 means 1st number is lesser than 2nd number i.e., parameter

0 means 1st & 2nd number both are equal

1 means 1st number is greater than 2nd number

Integer.compare(20, 10): returns 1

Integer.compare(10, 20): returns -1

Integer.comapre(10, 10): returns 0

Double.compare(10.5, 9.5): returns 1

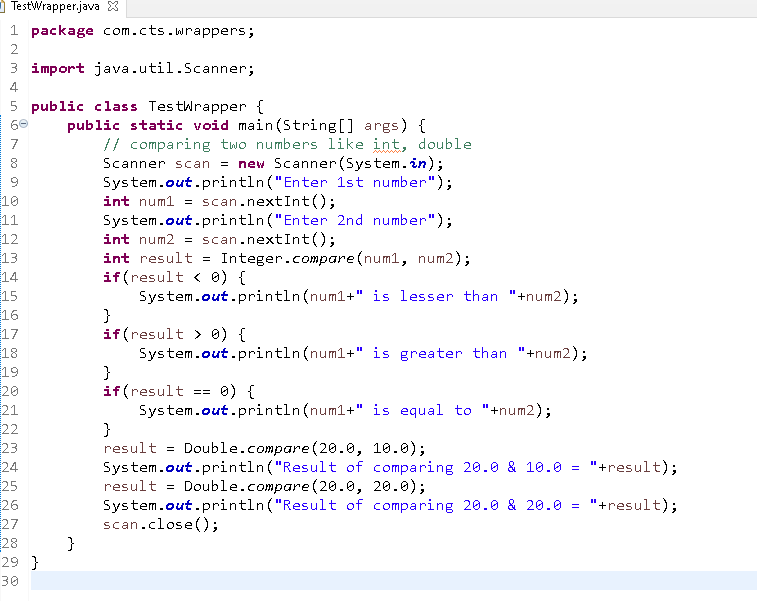
Double.compare(9.5, 10.5): returns -1

Double.compare(9.5, 9.5): returns 0

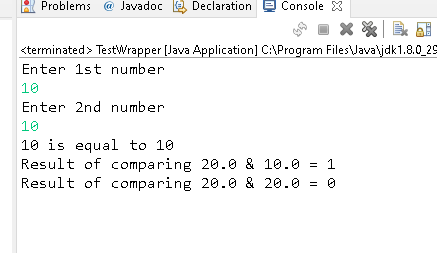
Note: compare method takes 2 parameters & it is present in all the number type wrapper classes like Byte, Short, Integer, Long, Double, Float

Note: compare method returns int value like -ve, 0, +ve based on that you can decide whether 2 numbers are equal, or lesser/greater than the one.

TestWrapper.java



Output:



String:

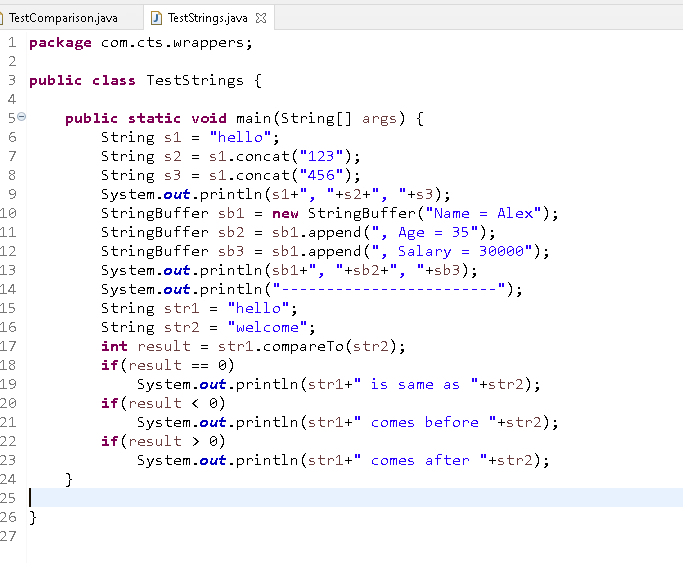
It creates immutable string objects, immutable means once string content is created it can’t be modified

It has many methods to work with strings like: concat, equals, equalsIgnoreCase, toUppercase, toLowercase, length, compareTo

StringBuffer:

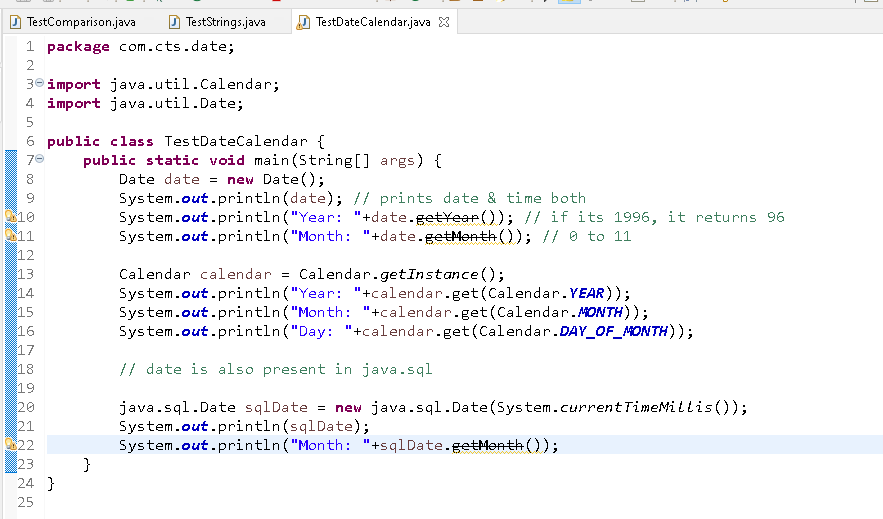
It creates mutable string object, content in string buffer can be modified.

It has many methods to work with strings like: append, insert, delete



Date, Calendar classes

Date is present in java.util as well as java.sql, it shows date & time both, but many of the methods give unexpected results as the Date classes were designed during 90’s, ex: People used to write year like 91, 92 for 1991, 1992, but for 2000 and afterwards these should be replaced with 00, 01, 02, and so on, but Date class were giving 91, 92 using the constant 1900, it means Date was designed keeping 1900 in mind, so for 1992 it must give 92, for that they subtract 1992 with 1900, but for 2000 onwards it started giving 100 instead of 00, hence many methods were deprecated as they were depending on 1900.



To replace these API’s Java introduced new Date & Time Api’s from Java 8 onwards like LocalDate, LocalTime, LocalDateTime

Collection Framework