\* ECW

1\* == vs equals()

2\* immutable , Custom immutable class with a getter method

3\* Methods of object class

4\* Marker interface

5\* System.out.println meaning

6\* System.out = null; allowed or not? no

7\* Use of static----https://beginnersbook.com/2013/04/java-static-class-block-methods-variables/

8\* Serialization, how it works internally.

9\* How to create Singleton object. With multi threading

10\* Synchronization.

\* Hashmap and hashtable.

\* Sql joins

\* Employee: id, name, did, salary

\* Dept: did, name

\* Find emp with dname

\* Display dname with avg employee salary.

\* Wrapper class.

Wrapper classes are used to convert any data type into an object. The primitive data types are not objects; they do not belong to any class; they are defined in the language itself. Sometimes, it is required to convert data types into objects in Java.

Array second highest

\* Find given string can form palindrome or not.

\* :

\* Lifecycle of jsp

\* Forward vs redirect

OuterClass.InnerClass innerObject = outerObject.new InnerClass();

statuc-------OuterClass.StaticNestedClass nestedObject = new OuterClass.StaticNestedClass();

2-----------------------------------------------------------

Immutable class means that once an object is created, we cannot change its content. In Java, all the wrapper classes (like String, Boolean, Byte, Short) and

String class is immutable. We can create our own immutable class as well.

Following are the requirements:

• Class must be declared as final (So that child classes can’t be created)

• Data members in the class must be declared as final (So that we can’t change the value of it after object creation)

• A parameterized constructor

• Getter method for all the variables in it

• No setters(To not have option to change the value of the instance variable)

// An immutable class

public final class Student

{

final String name;

final int regNo;

public Student(String name, int regNo)

{

this.name = name;

this.regNo = regNo;

}

public String getName()

{

return name;

}

public int getRegNo()

{

return regNo;

}

}

3----------------------------------------

3 toString

equals

hashcode

getclass

clone

wait

notifyall

notify

4--------------------------------------------------------

Marker interface in Java is interfaces with no field or methods. In simple words, empty interface in java is called marker interface.

Example of marker interface is Serializable, Cloneable and Remote interface.

These are used to indicate something to compiler or JVM.

5---------------------------------------------------------------

System.out.println is a Java statement that prints the argument passed, into the System.out which is generally stdout.

System is a Class

out is a Variable

println() is a method

System is a class in the java.lang package . The out is a static member of the System class, and is an instance of java.io.PrintStream . The println is a method of java.io.PrintStream. This method is overloaded to print message to output destination, which is typically a console or file.

the System class belongs to java.lang package

6--------------------------------------------------------------------System.out.println(null);

There are 3 println methods in PrintStream that accept a reference type - println(char x[]), println(String x), println(Object x).

When you pass null, all 3 are applicable. The method overloading rules prefer the method with the most specific argument types, so println(Object x) is not chosen.

Then the compiler can't choose between the first two - println(char x[]) & println(String x) - since String is not more specific than char[] and vice versa.

If you want a specific method to be chosen, cast the null to the required type.

For example :

System.out.println((String)null);

9------------------------------------

Singleton Class in Java

2.4

In object-oriented programming, a singleton class is a class that can have only one object (an instance of the class) at a time.

After first time, if we try to instantiate the Singleton class, the new variable also points to the first instance created. So whatever modifications we do to any variable inside the class through any instance, it affects the variable of the single instance created and is visible if we access that variable through any variable of that class type defined.

To design a singleton class:

Make constructor as private.

Write a static method that has return type object of this singleton class. Here, the concept of Lazy initialization in used to write this static method.

Normal class vs Singleton class: Difference in normal and singleton class in terms of instantiation is that, For normal class we use constructor,

whereas for singleton class we use getInstance() method (Example code:I). In general, to avoid confusion we may also use the class name as method name while defining this method (Example code:II).

Implementing Singleton class with getInstance() method

// Java program implementing Singleton class

// with getInstance() method

class Singleton

{

// static variable single\_instance of type Singleton

private static Singleton single\_instance = null;

// variable of type String

public String s;

// private constructor restricted to this class itself

private Singleton()

{

s = "Hello I am a string part of Singleton class";

}

// static method to create instance of Singleton class

public static Singleton getInstance()

{

if (single\_instance == null)

single\_instance = new Singleton();

return single\_instance;

}

}

// Driver Class

class Main

{

public static void main(String args[])

{

// instantiating Singleton class with variable x

Singleton x = Singleton.getInstance();

// instantiating Singleton class with variable y

Singleton y = Singleton.getInstance();

// instantiating Singleton class with variable z

Singleton z = Singleton.getInstance();

public class SingletonDesignPatternInMultiThreadedEnvironment {

    // Step 1: private static variable of INSTANCE variable

    private static volatile SingletonDesignPatternInMultiThreadedEnvironment INSTANCE;

    // Step 2: private constructor

    private SingletonDesignPatternInMultiThreadedEnvironment() {

    }

    // Step 3: Provide public static getInstance() method returning INSTANCE after checking

    public static SingletonDesignPatternInMultiThreadedEnvironment getInstance() {

        // synchronized block

        synchronized (SingletonDesignPatternInMultiThreadedEnvironment.class) {

            if(null == INSTANCE){

                INSTANCE = new SingletonDesignPatternInMultiThreadedEnvironment();

            }

            return INSTANCE;

        }

    }

}

Questions

Programs-fectorial

Febonacci

palindrom

swap

Concepts

Collection

Thread

Exception

String

Hibernate

Highq