

core java - Notepad

File Edit Format View Help

The automatic conversion of primitive data types into its equivalent Wrapper type is known as boxing and opposite operation is known as unboxing
Whenever we create objects it occupies space in the heap memory while the reference of that object creates in the stack.

AOT AND JIT in java

When we write a program in any programming language it requires converting that code in the machine-understandable form because the machine only understands the binary language. use cmd javac to convert Java-In-Time Compiler. The JIT compilation is also known as dynamic compilation

If the JIT compiler environment variable is properly set, the JVM reads the .class file (bytecode) for interpretation after that it passes to the JIT compiler for further process. After getting the bytecode, the JIT compiler transforms it into the native code (machine-readable code).

```
String s = "football" //saved in string constant
String p = new String("football"); //saved in heap mem
s.equals(p)--true //equals check for content
s == p false//bcz address indexes r from diff
```

When we use the == operator for s1 and s2 comparison, the result is true as both have the same addresses in the string constant pool.

```
String s1 = new String("GEEKS");
String s2 = new String("GEEKS"); //in heap everytime new instance will get created hence address is diff
String myStr1 = "Hello";
String myStr2 = "Hello"; //in string pool both var will refer to same mem loc
System.out.println(s1 == s2); //false
System.out.println(s1.equals(s2)); //true
System.out.println(myStr1 == myStr2); //true
System.out.println(myStr1.equals(myStr2)); //true
```

Serialization in Java is a mechanism of writing the state of an object into a byte-stream.

SerialVersionUID is unique for all class and it represent metadata for that class.

Suppose class has few fields, from that SerialVersionUID will get generated.

singleton scope

Bean Scopes refers to the lifecycle of Bean that means when the object of Bean will be instantiated, how long does that object live, and how many objects will be created for that bean throughout. Basically, it controls the instance creation of the bean and it is managed by the spring container.

Singleton: Only one instance will be created for a single bean definition per Spring IoC container and the same object will be shared for each request made for that bean.

Ln 80, Col 1

100% Windows (CRLF)



core java - Notepad
File Edit Format View Help

singleton scope

Bean Scopes refers to the lifecycle of Bean that means when the object of Bean will be instantiated, how long does that object live, and how many objects will be created for that bean throughout. Basically, it controls the instance creation of the bean and it is managed by the spring container.

~~Singleton: Only one instance will be created for a single bean definition per Spring IoC container and the same object will be shared for each request made for that bean whenever a new request is made for that bean, spring IOC container first checks whether an instance of that bean is already created or not. If it is already created, then the IOC container returns the same instance otherwise it creates a new instance of that bean only at the first request. By default, the scope of a bean is a singleton.~~

A composite key is made by the combination of two or more columns in a table that can be used to uniquely identify each row in the table when the columns are combined uniqueness of a row is guaranteed, but when it is taken individually it does not guarantee uniqueness, or it can also be understood as a primary key made by the combination of two or more attributes to uniquely identify every row in a table.

palindrom

```
string s ="madam"  
reverse string using stringbuffer/builder functions and then compare..if same then palindrom
```

The IoC container is responsible to instantiate, configure and assemble the objects. The IoC container gets informations from the XML file and works accordingly.

abstraction is defined as hiding the unnecessary details (implementation) from the user and to focus on essential details (functionality). It increases the efficiency and thus reduces complexity.
abstract method do not have a body (implementation), they just have a method signature (declaration). The class which extends the abstract class implements the abstract methods.

multithreading-- multiple thread will do work parallel. threads can b created by extending thread class or by implementing interface it doesn't block the user because threads are independent and you can perform multiple operations at the same time
new-active-run-wait-sleep-

suppose 1 thread is waiting to read Q elements but there r no element at presn. it will release lock and another thread ll get execute.

on same thread we cant use start method twice..1 tread can b started only once

when we ll use multithreading we hv to override equals() and hashCode(). if we ll not override as per our requirement then it will take by

Ln 80, Col 1

100%

Windows (CRL)



core java - Notepad
File Edit Format View Help

multithreading-- multiple thread will do work parallel. threads can b created by extending thread class or by implementing interface new-active-run-wait-sleep-
suppose 1 thread is waiting to read Q elements but there r no element at presn. it will release lock and another thread ll get execute.

on same thread we cant use start method twice..1 tread can b started only once

when we ll use multithreading we hv to override equals() and hashCode(). if we ll not override as per our requirement then it will take by default object class hashCode method, internal hashCode mechanism will happen and on that basis values will stored in bucket and its address ref also.
but if we r doing emp e =new emp() everytime then new ref will get created and equals() method never get satisfied. so its better to override those methods.

default scope of bean--only 1 instance of spring bean will b created for spring container.

abstraction is achieved by interfaces and abstract classes.

<https://www.geeksforgeeks.org/create-immutable-class-java/>
<https://www.javatpoint.com/how-to-create-immutable-class>

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

The class must be declared as final so that child classes can't be created.

Data members in the class must be declared private so that direct access is not allowed.

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 should initialize all the fields performing a deep copy so that data members can't be modified with an object reference.

Deep Copy of objects should be performed in the getter methods to return a copy rather than returning the actual object reference

There should be no setters or in simpler terms, there should be no option to change the value of the instance variable.

```
public final class Employee
{
final String pancardNumber;
public Employee(String pancardNumber)
{
this.pancardNumber=pancardNumber;
```

Ln 80, Col 1 100% Windows (CRLF)



Immutable class in java means that once an object is created, we cannot change its content. In Java, all the wrapper classes (like Integer, Boolean, Byte, Short) and String class is immutable.
The class must be declared as final so that child classes can't be created.
Data members in the class must be declared private so that direct access is not allowed.
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 should initialize all the fields performing a deep copy so that data members can't be modified with an object.
Deep Copy of objects should be performed in the getter methods to return a copy rather than returning the actual object reference.

There should be no setters or in simpler terms, there should be no option to change the value of the instance variable.

```
public final class Employee
{
final String pancardNumber;
public Employee(String pancardNumber)
{
this.pancardNumber=pancardNumber;
}
public String getPancardNumber(){
return pancardNumber;
}
}
public class ImmutableDemo
{
public static void main(String ar[])
{
Employee e = new Employee("ABC123");
String s1 = e.getPancardNumber();
System.out.println("Pancard Number: " + s1);
}
}
```

The instance variable of the class is final i.e. we cannot change the value of it after creating an object.
The class is final so we cannot create the subclass.

There is no setter methods i.e. we have no option to change the value of the instance variable.

