1.Wipro Questions 21-07-2023:

=====================

1.Convert HashMap to ArrayList ?

Map<Integer, String>map=new HashMap<>();

map.add(1, "Laxman");

map.add(2, "Aditya");

List<String>list =new ArrayList<>(map.values());

list.forEach(System.out::println);

========================================================================

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class HashMapToArrayListExample {

public static void main(String[] args) {

// Create a HashMap

Map<String, Integer> hashMap = new HashMap<>();

hashMap.put("Alice", 25);

hashMap.put("Bob", 30);

hashMap.put("Charlie", 22);

hashMap.put("David", 28);

// Convert HashMap values to ArrayList

List<Integer> list = new ArrayList<>(hashMap.values());

// Display the ArrayList

System.out.println("ArrayList: " + list);

list .forEach(System.out::println);

}

}

========================================================================

How can Create a Custom immutable class ?

pulic final class Employee{

final String panCarsNumber;

public Employee(String panCardNumber){

this.panCardNumber = panCardNumber;

}

public String getPancrdNumber(){

return panCardNumber;

}

}

public class ImmutableDemo{

public static void main (string[] args){

Employee e = new Employee("abc123");

String s1 = e.getEmployee();

System.out.peintln("PanCrdNumber"+s1);

}

}

========================================================================Using lambda expression and functional interface to write a program of Square Number ?

@functionalInterface

interface PrintNumber{

public void print(int num1);

}

public class SquareNumber{

public static void main(String[] args){

PrintNumber p = (n)->System.out.println(n\*n);

p.print(5);

}

}

========================================================================

How to Save String in String constant pool ?

public class StringConstatntDemo{

public static void main(String[] args){

String s1 = "Hello";

String s2 = new String("Hello").intern();

String s3 = String.valueOf("Hello");

System.out.println(s1 == s2); //output => true

System.out.println(s1 == s3); //output => true

}

}

========================================================================5.In Singleton Design pattern how can control Multiple thread ?

with lazy initialization with with double checked locking :

========================================

public class Singleton{

private static volatile Singleton instance;

private Singleton(){}

public static Singleton getInstance(){

if(instance == null){

sinchronized(Singleton.class){

if(instance == null){

instance = new Singleton();

}

}

}

return instance;

}

}

========================================================================

How can get byte Stream in object in Synchronized ?

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.io.ObjectOutputStream;

public class SynchronizedSerializationExample {

private Object objectToSerialize;

public SynchronizedSerializationExample(Object objectToSerialize) {

this.objectToSerialize = objectToSerialize;

}

public synchronized byte[] getSerializedBytes() throws IOException {

ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();

ObjectOutputStream objectOutputStream = new ObjectOutputStream(byteArrayOutputStream);

objectOutputStream.writeObject(objectToSerialize);

objectOutputStream.close();

return byteArrayOutputStream.toByteArray();

}

}

========================================================================

String Why immutable ?

In object-oriented programming, the immutable string or objects that cannot be modified once it is created. But we can only change the reference to the object. We restrict to change the object itself. The String is immutable in Java because of the security, synchronization and concurrency, caching, and class loading. The reason of making string final is to destroy the immutability and to not allow others to extend it.

The String objects are cached in the String pool, and it makes the String immutable. The cached String literals are accessed by multiple clients. So, there is always a risk, where action performs by one client affects all other clients. For example, if one client performs an action and changes the string value from Pressure to PRESSURE, all remaining clients will also read that value. For the performance reason, caching of String objects was important, so to remove that risk, we have to make the String Immutable.

========================================================================

What are the Design Pattern implemented your project ?

1.Creational Design Pattern

Factory Pattern

Abstract Factory Pattern

Singleton Pattern

Prototype Pattern

Builder Pattern.

2. Structural Design Pattern

Adapter Pattern

Bridge Pattern

Composite Pattern

Decorator Pattern

Facade Pattern

Flyweight Pattern

Proxy Pattern

3. Behavioral Design Pattern

Chain Of Responsibility Pattern

========================================================================

List out Predefined Functional Interface ?

Built-in Functional Interfaces in Java 8 are:

Predicate

Consumer

Supplier

Function

========================================================================

How may ways Create Object in Java ?

1.New Keyword

2.Factory method

3.Clone method

4.Reflection api

5.Deserialization

========================================================================

How can you Create String in Java ?

There are two ways to create String object:

By string literal

By new keyword

1) String Literal

String s="welcome";

Each time you create a string literal, the JVM checks the "string constant pool" first. If the string already exists in the pool, a reference to the pooled instance is returned. If the string doesn't exist in the pool, a new string instance is created and placed in the pool. For example:

String s1="Welcome";

String s2="Welcome";//It doesn't create a new instance

Java String

In the above example, only one object will be created. Firstly, JVM will not find any string object with the value "Welcome" in string constant pool that is why it will create a new object. After that it will find the string with the value "Welcome" in the pool, it will not create a new object but will return the reference to the same instance.

========================================================================

What are Java 8 Features ?

=================

Java 1.8 Features

=================

1) Interface changes

1.1 ) Default Methods

1.2 ) Static Methods

2) Functional Interfaces (@FunctionalInterface)

2.1 ) Predicate & BiPredicate

2.2 ) Consumer & BiConsumer

2.3 ) Supplier

2.4 ) Function & BiFunction

3) Lambda Expressions

4) Method References & Constructor References

5) \*\*\*\*\*\* Stream API \*\*\*\*\*\*\*\*

6) Optional class (to avoid null pointer exceptions)

7) Spliterator

8) StringJoiner

9) forEach ( ) method

10) Date & Time API11) Nashron Engine

12) I/O Stream Changes (Files.lines(Path p))

13) Base64 Encoding & Decoding

========================================================================

Difference Between Spring and Spring Boot ?



========================================================================

How can create Spring Application and SpringBoot ?

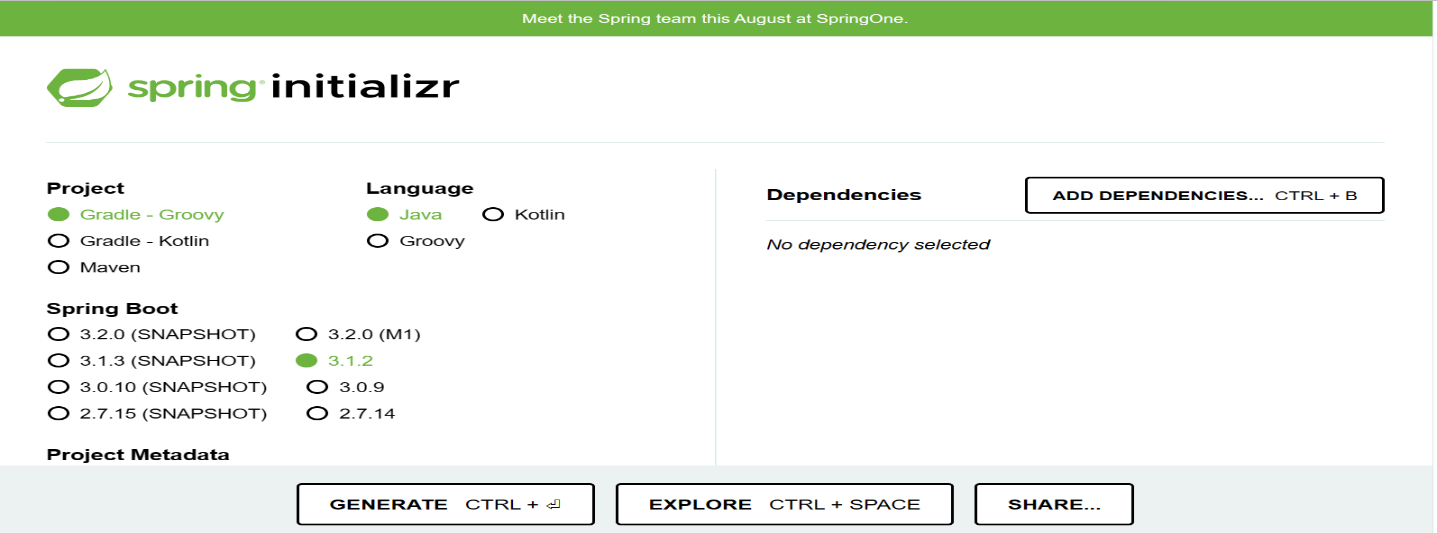
Creating a Spring Boot Project

Following are the steps to create a simple Spring Boot Project.

**Step1:** Open the Spring initializr [https://start.spring.io](https://start.spring.io/).

**Step 2:** Provide the **Group** and **Artifact** name. We have provided Group name **com.javatpoint** and Artifact **spring-boot-example**.**Step**

**3:** Now click on the **Generate** button.



**What is actuator ?**

**Spring Boot Actuator**

**Spring Boot Actuator** is a sub-project of the Spring Boot Framework. It includes a number of additional features that help us to monitor and manage the Spring Boot application. It contains the actuator endpoints (the place where the resources live). We can use **HTTP** and **JMX** endpoints to manage and monitor the Spring Boot application. If we want to get production-ready features in an application, we should use the S**pring Boot actuator.**

Spring Boot Actuator Features

There are **three** main features of Spring Boot Actuator:

**Endpoints**

**Metrics**

**Audit**

**Endpoint:** The actuator endpoints allows us to monitor and interact with the application. Spring Boot provides a number of built-in endpoints. We can also create our own endpoint. We can enable and disable each endpoint individually. Most of the application choose **HTTP**, where the Id of the endpoint, along with the prefix of **/actuator,**is mapped to a URL.

For example, the **/health** endpoint provides the basic health information of an application. The actuator, by default, mapped it to **/actuator/health**.

========================================================================

I dont want Default Server in SpringBoot?

Exclude the embedded web server dependency in your build configuration. If you are using Maven, you can add the following exclusion to your pom.xml

<properties>

<!-- ... other properties ... -->

<spring-boot-starter-web.version>2.5.4</spring-boot-starter-web.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

<version>${spring-boot-starter-web.version}</version>

<exclusions>

<exclusion>

<groupId>org.springframework.boot</groupId>

<! -- Replace 'spring-boot-starter-tomcat' with 'spring-boot-starter-jetty' or 'spring-boot-starter-undertow'

If you prefer Jetty or Undertow as the embedded web server. -->

<Artifact ID>spring-boot-starter-tomcat</artifact ID>

</exclusion>

</exclusions>

</dependency>

<! -- ... other dependencies ... -->

</dependencies>

========================================================================

17. Track Minds 10-8-2023

======================

28. What is defference between runnable and thread class wich one is better and why?

Implements Runnable interface is better because inherit any other class and loosely coupled and less memory is require whereas extending thread class can not inherit any other class and it is tightly coupled and more memory require that is the main reason .

Class MyThread extends Thread {

Public void run () {

For (int i=0; i<5; i++) {

System.out.println (“Child Thread ….”);

}

}

Class mainThread {

Public static void main (String [] args) {

MyThread mt = new Mythread ()

mt.start ();

For (int i=0; i<5; i++) {

System.out.println (“Main Thread…..”);

}

}

}}

Class RunnableMyThread implements Runnable {

Public void run ( ) {

For (int i=0; i<5; i++) {

System.out.println (“Child Thread…….”);

}

}

Class mainThread {

Public static void main(String [ ] args){

MyThread rt = new MyThread ( );

Thread t = new Thread ( rt );

t.start();

for (int i=0; i<5; i++) {

System.out.println(“Main Thread….”);

}

}

}

}

========================================================================