**Core Java:-**

**Why Java is not 100% Object Oriented ?**

It uses the Primitive Datatypes namely

Boolean,byte,char,int,float,double.

To make them OO we uses Wrapper classes

**Why pointers are not used in Java?**

They are unsafe, It increases the complexity of the program. Since JVM is responsible for memory allocation. If it uses the direct allocation of the address it leads to the memory leakage.

**What is JIT Compiler?**

**JIT** compiler is responsible for performance of the java application at runtime. It does because interpreter will take time line by line. JIT compiles bunch of code which is already repeated.

**Why String immutable in Java?**

Can’t modify once Object is created. Because it uses the String pool which is shared reference can be changed from anywhere.

Second reason is String is used in Database Connection, Passwords, file system.

No one can change the reference.

**Interview Questions?**

**What is platform independent?**

We can write the java program in one operating system and

we can run on any other operating system

**Tell me about your project?**

Tell me about yourself?

Self Introduction ?

Roles and Responsibilities in your Project?

Introduce yourserlf?

Hi I am Satya, I am working as a Full stack developer having 7+ years of Experience

working with various Technologies Spring Boot, Microservices, Spring ,Core Java, HTML

Currently, I am Associated with GAP Which is US based company

Dealing with Inventory and Control Management.

In our project we are 8 memebers team involves into Development,Testing,Integration

Deployment of the whole Project.

Inventory involves into multiple markets like US,JAPAN,CANADA and multiple stores

offline and online stores like nodes,

We deal with the ATP(Available to Promise) and ATS( Available to Shilpment).

Our Application monitors the Inventory in stores and send updates of the inventory

to the Downstreams (Web applications).

Day to Day Activities:-

We follow the Agile methodology Ideally sprint falls for 15 days. It involes Requirements gathering and analysis.

Once We assign the story points. User story will move to the Ready to pick and We as a team involves into Development and Testing.

We use 3 environements for deployment.

Test

Stage

and Prod.

**Tell me some java features ?**

simple to use, robust(Self Exception Handling mechanism),

secure,Object oriented, portable(Can run on any other operting System).

**Explain about Java OOps concepts?**

class,object,inheritance,polymorphism,encapsulation

abstraction

**What is inheritance?**

acquiring properties of one class into another class is inheritance

class A extends B

**What is a Class and some example?**

Blueprint which has some properties like Variables and methods

We create objects for accessing these properties

**Method over loading and over riding?**

methodname is same but parameters are diff

methodname and paramerts are same

**Why String class is immutable?**

cant change the value of String

String is used in security purpose like passwords files storage.

**What is abstraction**

Hiding irrelevant data from the user is abstraction

**Collections which collections did u use in your project?**

Collection interface

ArrayList

HashMap

HashSet

**Are u following agile methodology in your company?**

yes

**Sprint cycle ?**

15 days we follow in project

User stories we assign with in the team

Development and Testing involved

Daily Scrum meeting will be happen

**Difference between ArrayList and Linkedlist?**

Both uses the List Interface

Arraylist is best for update elements

linkedlist is best for inserting elements

**Explain Hashmap? diff between Map and Set ?**

Map uses Key value pair Set unique elements no duplication allows

**Explain about Exception Handling in java ?**

handing abrupt termination of a program is exception handling

two types 1. Checked Exception 2. Unchecked Exception

**Constructor**

constructor is also a method name having sameclass name used to initialize the variables

two types of constructors

default constuctor

parameterized constructor

**Abstract class and Interface**

having abstract methods and concrete methods is Abstract class

having only abstract methods is interface

**Does Java supports Multiple Inheritance?**

No. Acquiring properties from multiple classes ambiguity for the same method from two classes.

**Marker Interface:-** Marker interface does not have any data member and member functions.

Any empty Interface is a marker interface.

Ex:- Serializable, Clonable

These interfaces tells the jvm to do some particular Implementation.

**Can We override the Private method in Java:-**

Cant over ride private and static method.

If we override the static method it will treat as a subclass method.

**Does finally always execute in Java?**

Yes but in two cases

1. System.exit()
2. System crashes happen in any of the circumstances

**What method does an Object class have ?**

Java.lang.Object

clone()

equals()

finalize() ---It is called by garbage collector

getClass()

hashCode()

toString() method, notify(), notifyAll()

**How can I Make my Class Immutable?**

Make that class final

All the mutable fields are final

Fields as private

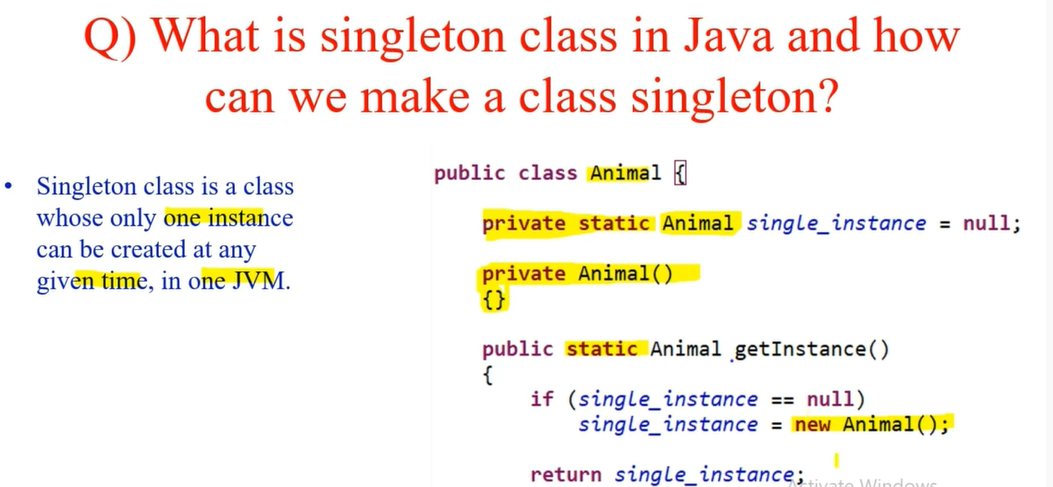
Initialize all the fields via constructor

Don’t provide the setters only getters

**What is singleton how to make singleton class?**

Private Constructor{}

Static method instance to get that object.



**Interface:-**

It is a mechanism to achieve abstraction. It has Abstraction methods

**Abstract class:-** class which has abstract methods and concrete method

**Java 8 Features:-**

Predicate:- Predicate is an Functional interface which has only one abstract method. It uses to check some Boolean condition

Predicate<String> checkLenght=s->s.length()>=5;

checkLength.test(“SattiBabu”);

Predicate<String> checkEvenLenght=s->s.length()%2 ==0;

Predicate Joining:-

Predicate combine with multiple predicates.

checkLength.and(checkEvenLength).test(“hljflajflaj”);

checkLength.or(checkEvenLenth).test(“jalfjlsafjl);

checklength.nagate().test(“ljaljfl);

nagate will work with one input.

**Function**:- It is also an Functional Interface which has only one method

apply(T t);

why function if we have predicate. Predicate will give only true or false. Function will give some result based on the input not only Boolean value.

Returntype is not fixed.

Function will take 2 inputs

R apply(T t);

Function<Integer,Integer> squareValue =i->i\*I;

squareValue.apply(10);

**Functional Chaining**:-

Function<Integer,Integer> cube =i->i\*i\*i;

squareValue.andThen(cubeit).apply(2); ---64

squarevalue.compose(cubeit).apply(2);--16

**Consumer:-**

Consumer<Integer> consumerTest=i->Syso(“Squaring value”+i\*i);

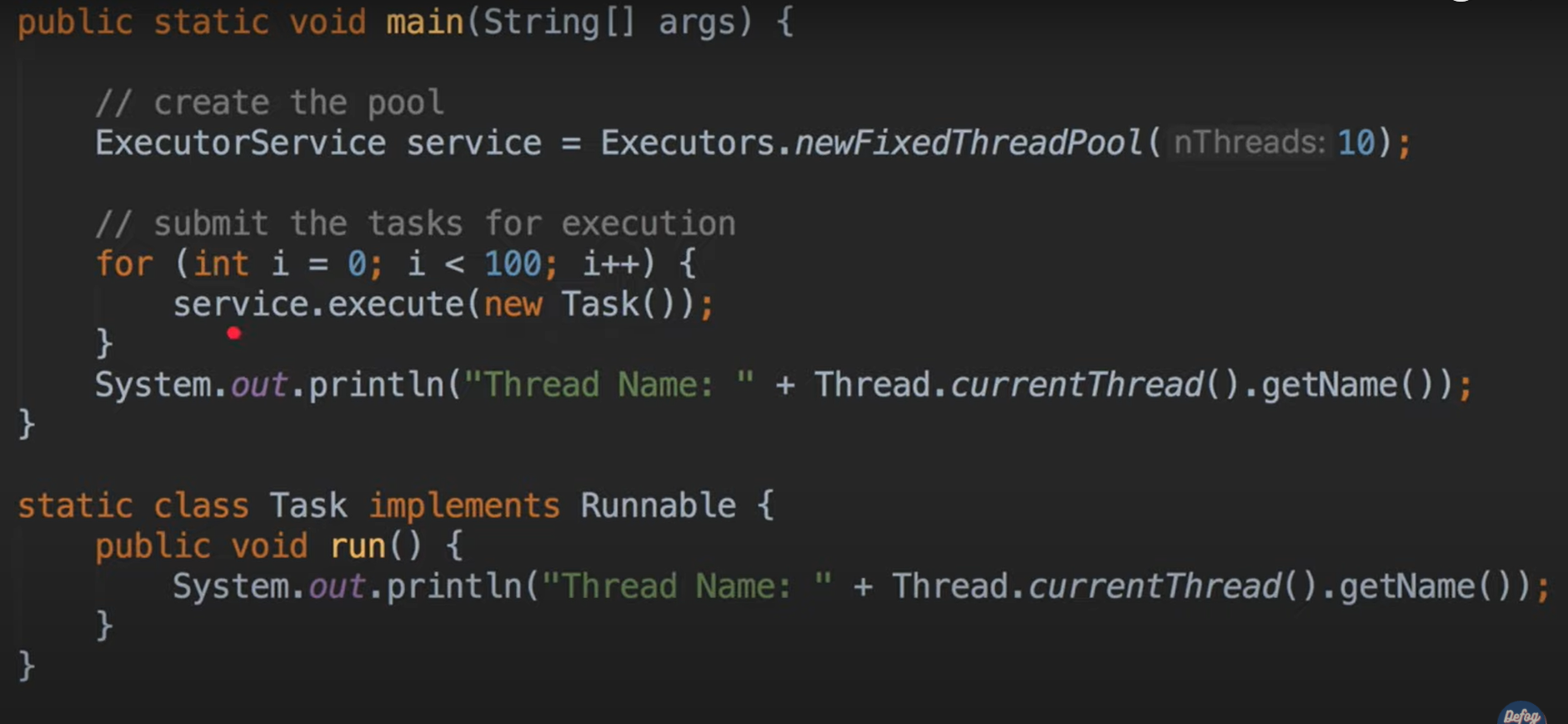
consumerTest.accept(5);

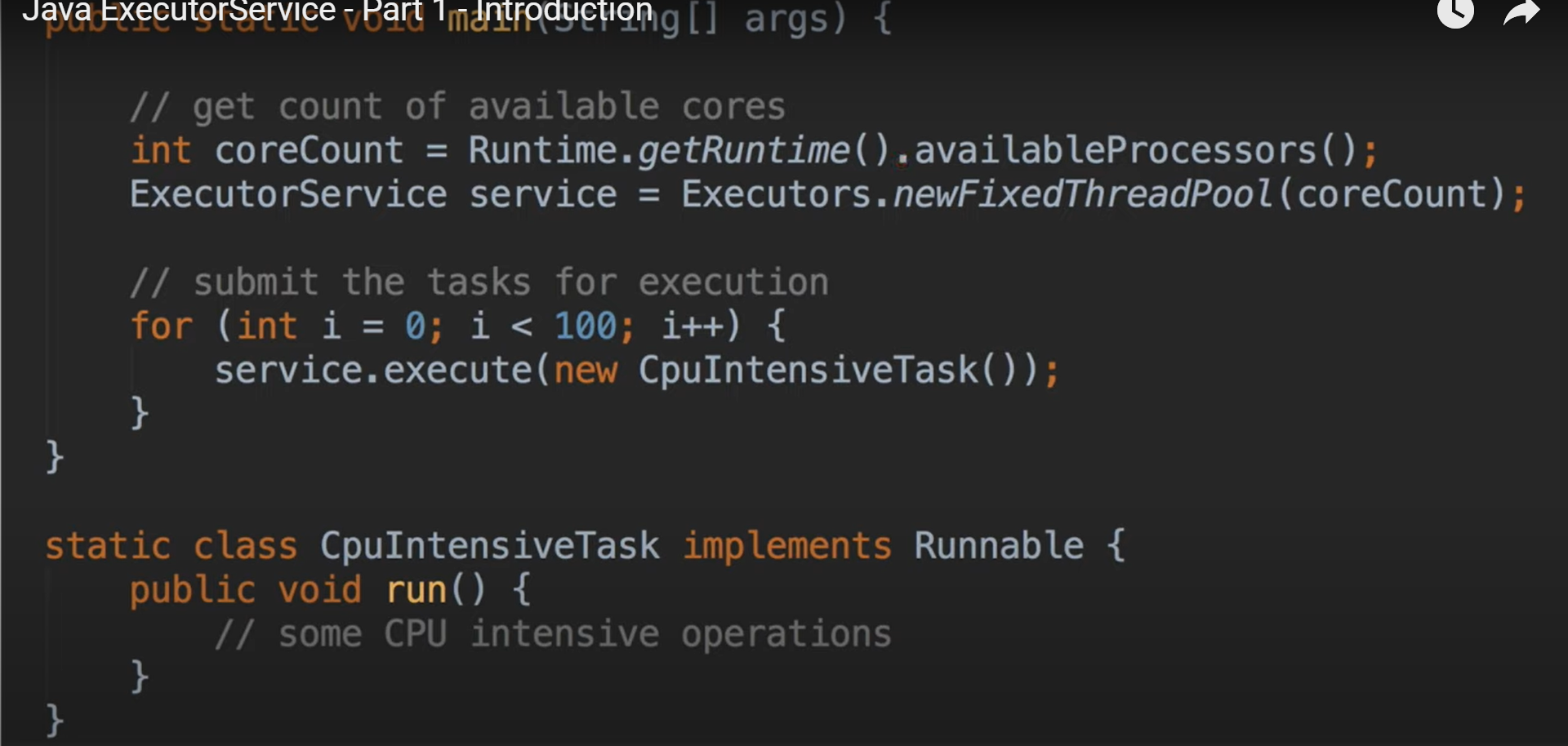
consumerTest.andThen(doubleMe).accept();

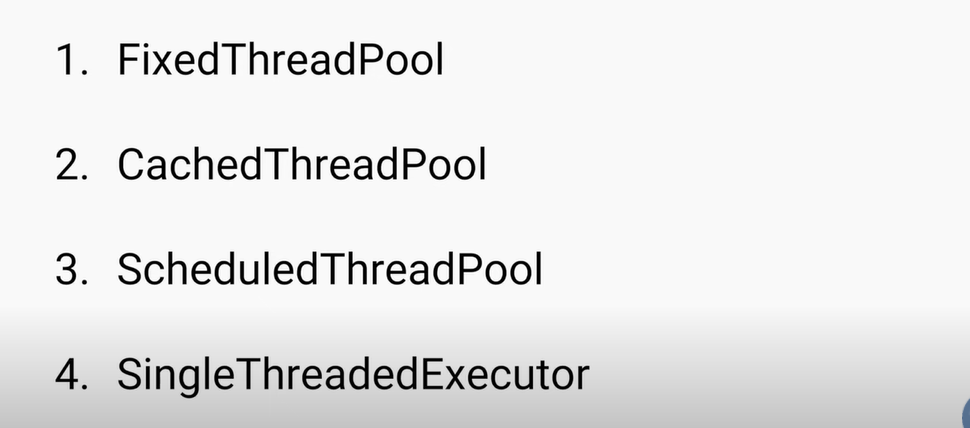
**Supplier:-**

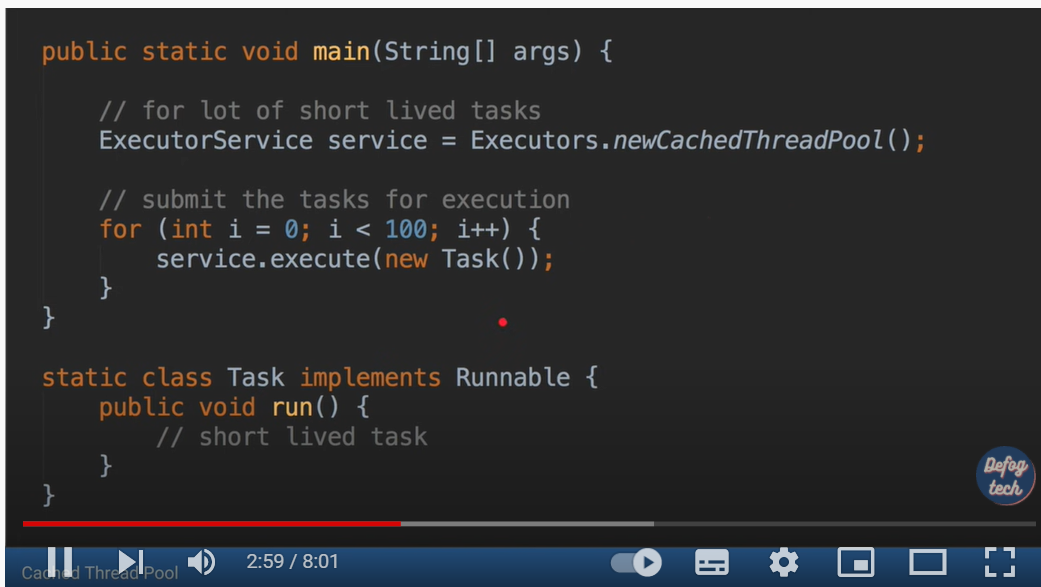
Supplier<String> supplierTest=()->return

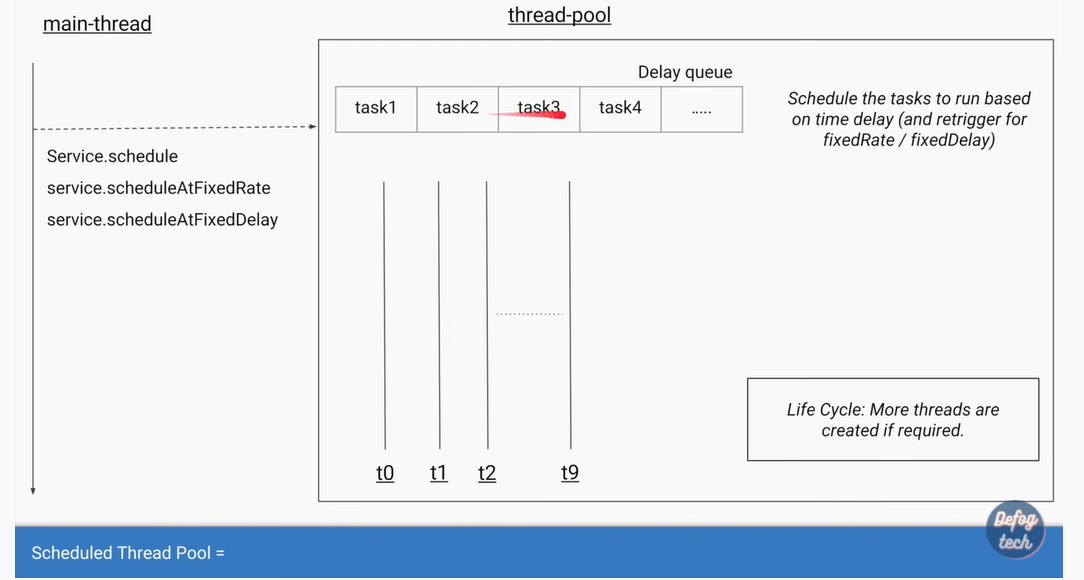
**MultiThreading:-**

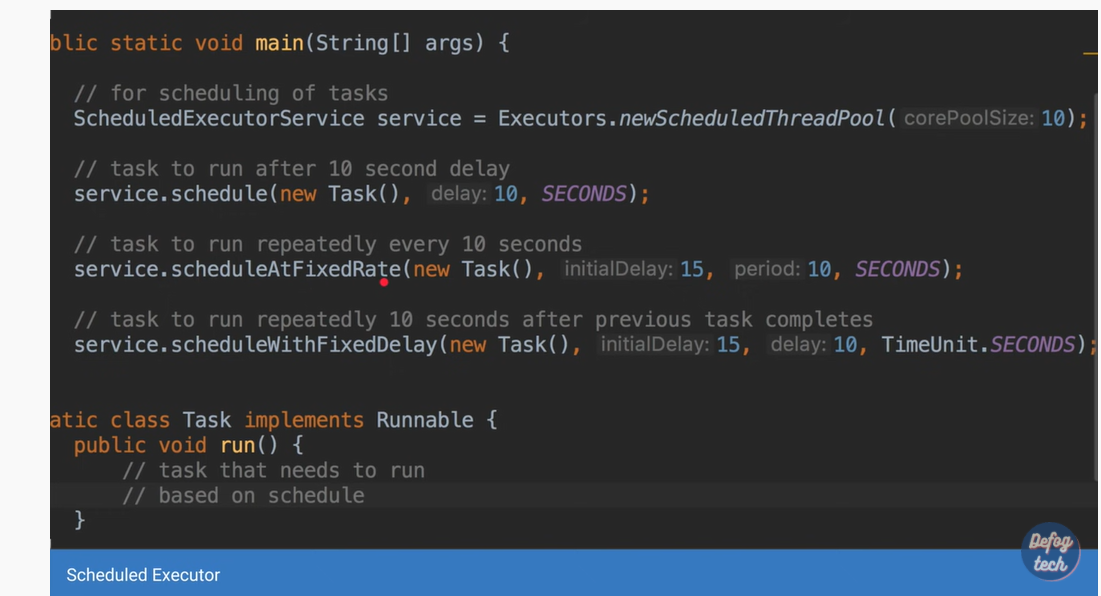


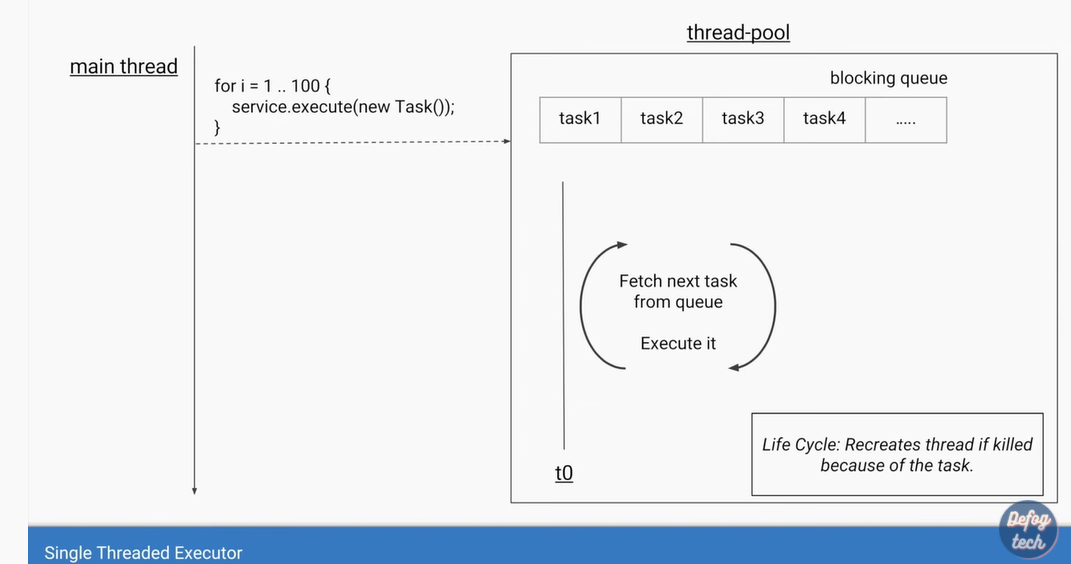


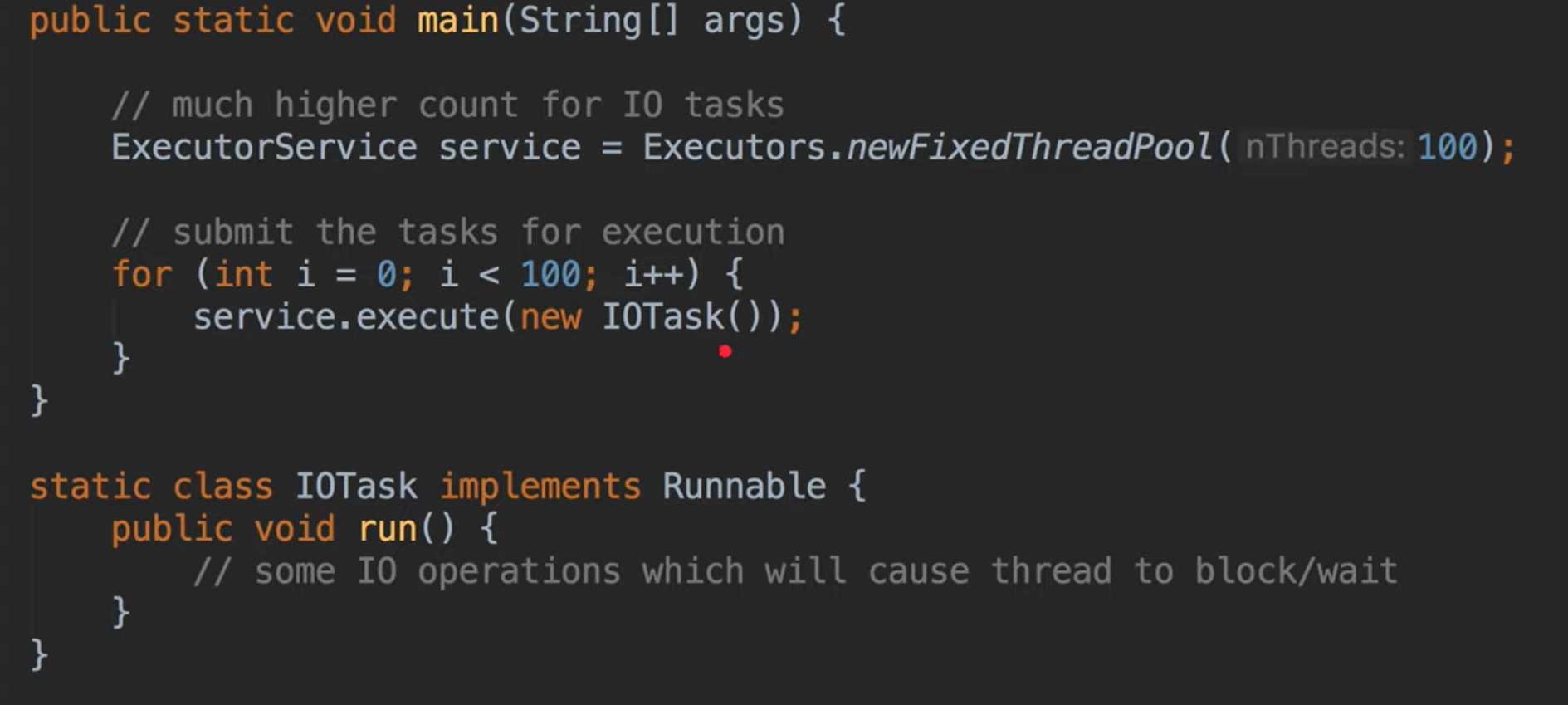












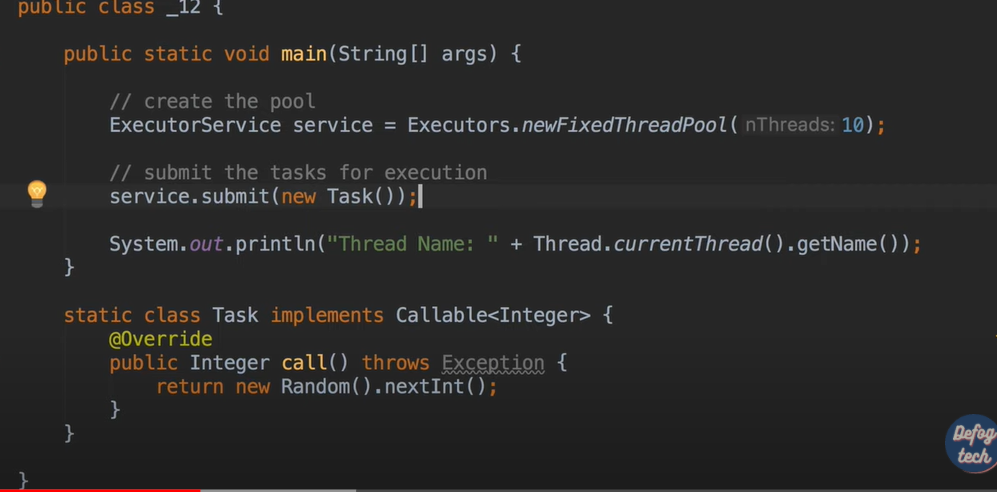
**Callable :- which is used to return some value from the run method instead of**

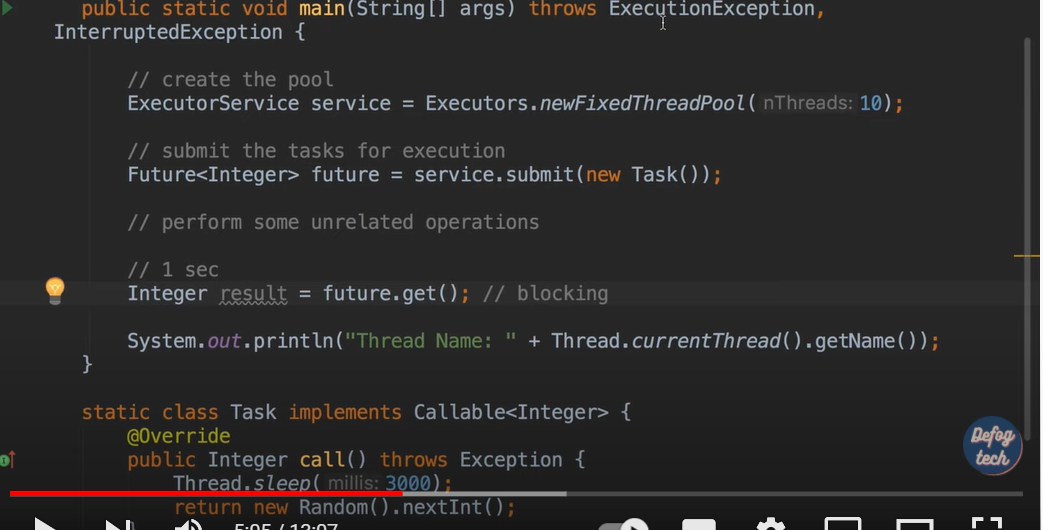
**Service.execute(**

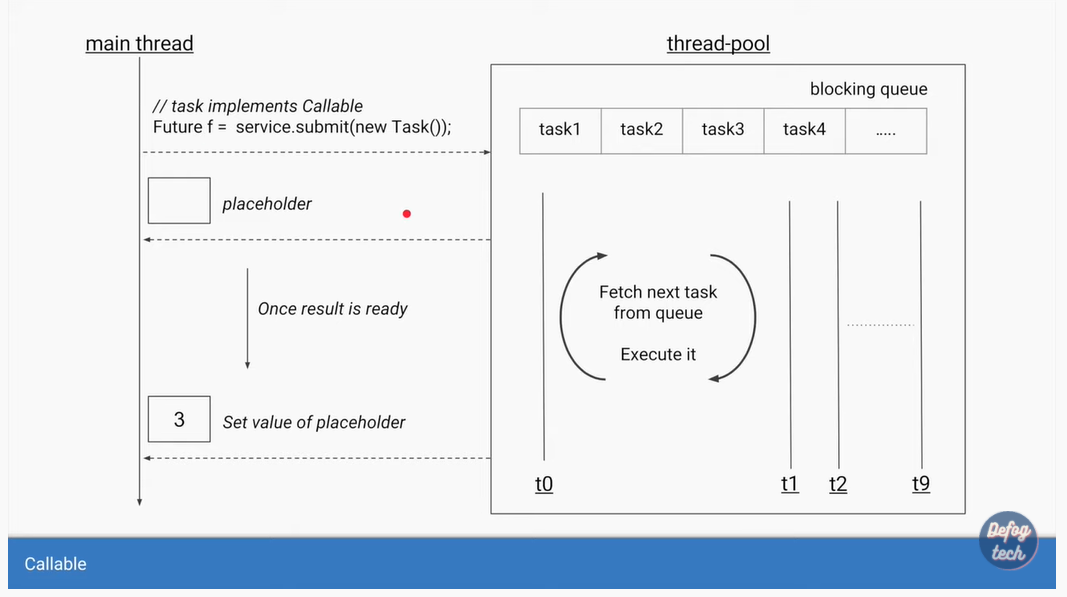
**Service.submit()**

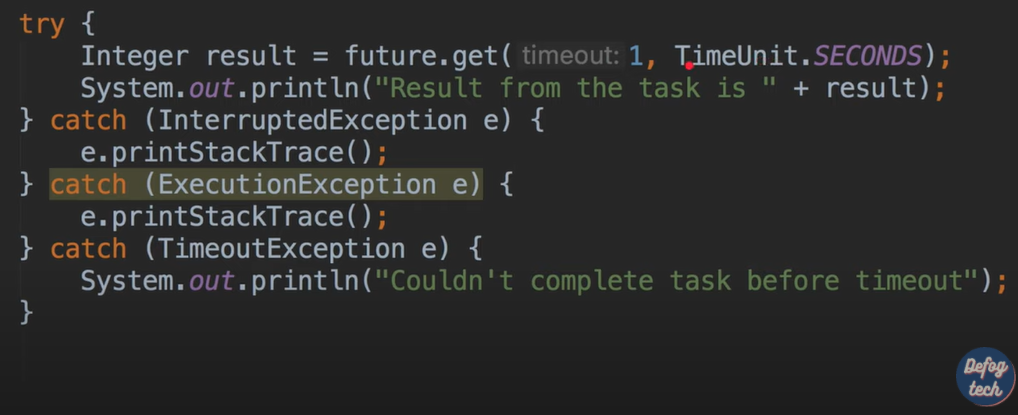
**Implements Callable<Integer>**

**Call method will be there**



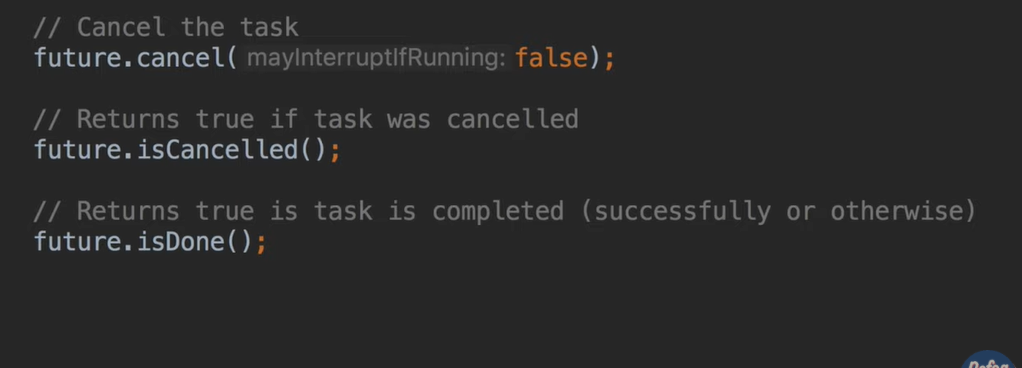






**Other methods in future**

**Future.cance()**



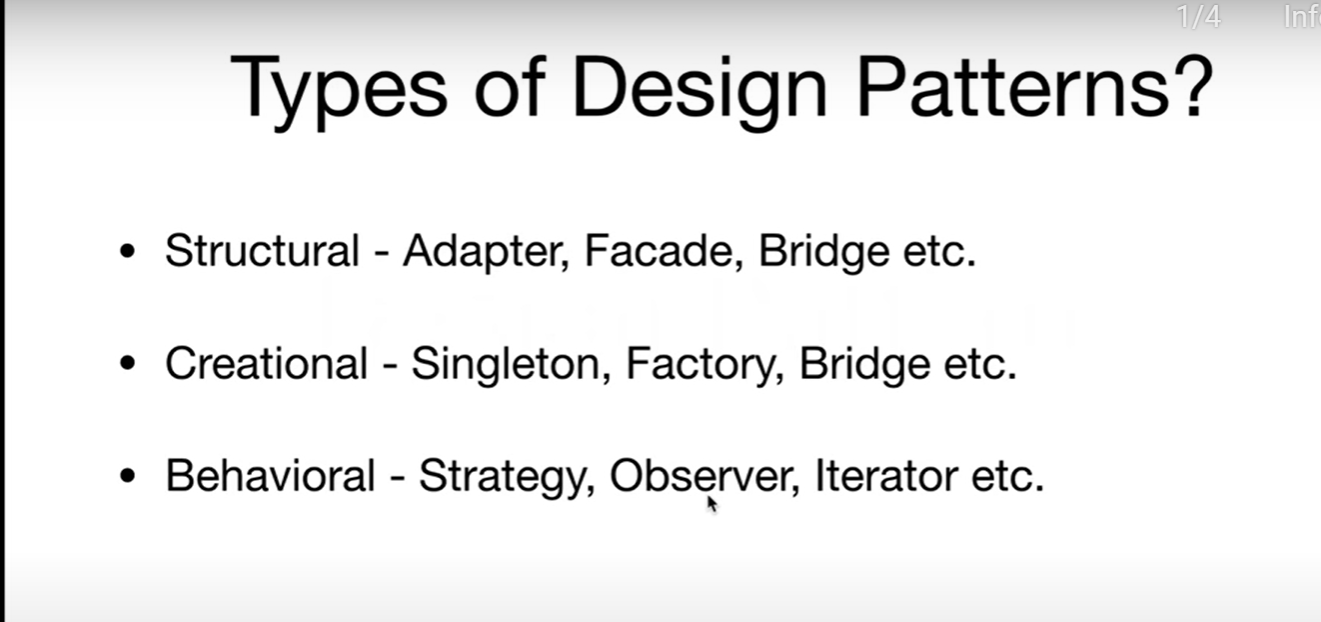
**Design Patterns:-**

Design Patterns:-

Design patterns is a common solution to the recurring problem. It suggests the reusability that gives robust and highly maintainable code. Solve common problems How to properly create a class, object, interaction between two objects,

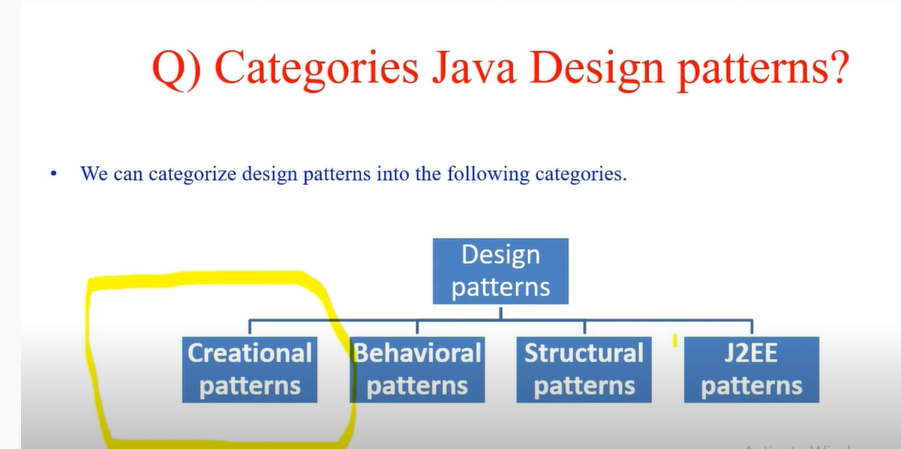
Loosely coupled code, reusable code.

4 types of Design patterns:-



Best Practices in Design Patterns:-

* Keep it Simple
* Excess of everything is bad
* Focus on loosely coupled code
* Design first code later



Creational Design Pattern:-

* 1. Singleton:- Restricts a class to instantiate multiple objects. One instance for the class exists in the virtual machine.

private Constructor

static variable for object instantiation

* 1. Factory:-Which is used when we have super class and multiple subclasses based on the input return one of the subclass.
  2. Abstract factory
  3. Builder
  4. Prototype Design Pattern

Structural Design Pattern:-

1. Adapter Pattern
2. Composite Pattern
3. Proxy Pattern
4. Flyweight Pattern
5. Bridge Pattern
6. Decorator pattern
7. Façade Pattern

Behavioral Design pattern:-

* 1. Template Method
  2. Chain of Responsibility pattern
  3. Mediator Pattern
  4. Observer Pattern
  5. Strategy Pattern
  6. State Pattern
  7. Visitor Pattern

J2EE Design patterns:-

1. DAO Design pattern
2. Dependency Injection pattern
3. MVC Pattern