1. Create a custom annotation called @Test which can be only applied on a method implying that the following method Is a test-case.

Ans🡪

**package** org.annotation.ex;

**import** java.lang.annotation.ElementType;

**import** java.lang.annotation.Retention;

**import** java.lang.annotation.RetentionPolicy;

**import** java.lang.annotation.Target;

**import** java.lang.reflect.Method;

@Retention(RetentionPolicy.***RUNTIME***)

@Target(ElementType.***METHOD***)

**@interface** Test{

String str();

}

**class** Hello{

@Test(str="This is Test Annotation ")

**public** **void** testcase() {}

}

**public** **class** Example1 {

**public** **static** **void** main(String[] args)**throws** Exception {

Hello h = **new** Hello();

Method m= h.getClass().getMethod("testcase");

Test ts= m.getAnnotation(Test.**class**);

System.***out***.println(ts.str());

}

}

Output:

This is Test Annotation

1. Build a custom annotation called @info, which can be used by developers on a class, a property, or method. The developer can provide the following information when using this annotation:
2. AuthorID: <<Developers ID>> -(Mandatory Input)
3. Author: <<Developers name>> - (Optional input)
4. Supervisor: <<Developers Supervisor>> -(Optional input)
5. Date: <<String Date>> -( Mandatory Input)
6. Time:<<String Time>> - (Mandatory Input)
7. Version:<<Numerical Version>> -( Mandatory Input)
8. Description: <<Description of the class, method, or property>> - (Optional Input)

Ans🡪

**package** org.annotation.ex;

**import** java.lang.reflect.Method;

**import** javax.sound.midi.MidiDevice.Info;

**public** **class** Example2 {

**public** **static** **void** main(String[] args) **throws** NoSuchMethodException, SecurityException {

second h =**new** second();

h.display();

Method m = h.getClass().getMethod("display");

info st= m.getAnnotation(info.**class**);

System.***out***.println("Author Id :" +st.ID());

System.***out***.println("Author Name :" +st.name());

System.***out***.println("Supervisor Name :" +st.supervisor());

System.***out***.println("Date :" +st.date());

System.***out***.println("Time :" +st.time());

System.***out***.println("Version :" +st.version());

}

}

**package** org.annotation.ex;

**import** java.lang.annotation.Documented;

**import** java.lang.annotation.Retention;

**import** java.lang.annotation.RetentionPolicy;

@Documented

@Retention(RetentionPolicy.***RUNTIME***)

**@interface** info{

**int** ID();

String name();

String supervisor();

String date();

String time();

**int** version();

}

**public** **class** second {

@info(ID=102, name="Chaitali", supervisor="capg",date="18/01/2022",time="17:02:23 hrs",version=12)

**public** **void** display()

{

System.***out***.println("Hello @info annotation");

System.***out***.println();

}

}

Output:

Author Id :102

Author Name :Chaitali

Supervisor Name :capg

Date :18/01/2022

Time :17:02:23 hrs

Version :12

1. Create a custom annotation called @Excute to be applied on methods. Placing the @Excute method on a method implies that method should be invoked using Reflection API. The annotation @Execute should have an optional property “sequence” which can be given values such a 1,2,3… in the order of priority. In case the sequence property is not used the API may invoke methods in a random order.

E.g.

Class MyClass{

@Execute(Sequence=2)

Public void myMethod1(){}

@Execute(Sequence=1)

Public void myMethod2(){}

@Execute(Sequence=3)

Public void myMethod3(){}

}

Ans🡪 **package** org.annotation.ex;

**import** java.lang.reflect.Method;

**public** **class** MyClass2 {

**public** **static** **void** main(String[] args) {

MyClass my= **new** MyClass();

Method[] methods = my.getClass().getMethods();

**for**(Method method : methods) {

Execute exe = method.getAnnotation(Execute.**class**);

**if**(exe!= **null**) {

**try** {

method.invoke(my);

}

**catch**(Exception e) {

e.printStackTrace();

}

}

}

}

}

**package** org.annotation.ex;

**import** java.lang.annotation.ElementType;

**import** java.lang.annotation.Retention;

**import** java.lang.annotation.RetentionPolicy;

**import** java.lang.annotation.Target;

@Target(value = ElementType.***METHOD***)

@Retention(value = RetentionPolicy.***RUNTIME***)

**@interface** Execute{

**int** Sequence();

}

**public** **class** MyClass {

@Execute(Sequence=2)

**public** **void** method1()

{

System.***out***.println("This is method 1");

}

@Execute(Sequence=1)

**public** **void** method2() {

System.***out***.println("This is method 2");

}

@Execute(Sequence=3)

**public** **void** method3()

{

System.***out***.println("This is method 3");

}

}

Output:

This is method 2

This is method 1

This is method 3