CORE JAVA

ANNOTATION ASSIGNMENT

1. Create a custom annotation called @Test which can be only applied on a method implying that the following method is a test-case. (Is it possible to restrict the annotation to be applied only on a non-static method?)
   Code:-

CustomAnnotationTest.java

```java
package annotation;
import java.lang.annotation.*;
import java.lang.reflect.*;
 @Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
@interface Test
{
String TestCase() ;
}
class test1
{
 @Test(TestCase = "Custom Annotation @Test")
public void Hello()
{
System.out.println("Lets try annotation");
}
}
public class CustomAnnotationTest
{
public static void main(String[] args) throws Exception
{
test1 t = new test1();
Method annot = t.getClass().getMethod("Hello");
Annotation annotation = annot.getAnnotation(Test.class);
Test ts = (Test)annotation;
System.out.println( ts.TestCase());
}
}
```

Problems | Javadoc | Declaration | Console | Git Staging

```
<terminated> CustomAnnotationTest [Java Application] C:\Users\MBALKRIS\.p2\poo
Custom Annotation @Test
```

2. Build a custom annotation called @Info, which can be used by developers on a class, a property, or a method. The developer can provide the following information when using this annotation:
   a) AuthorID:<<Developers ID>> - (Mandatory Input)
   b) Author:<<Developers name>> - (Optional Input)
   c) Supervisor:<<Developers Supervisor>> - (Optional Input)
   d) Date:<<"String Date">> - (Mandatory Input)
   e) Time:<<"String Time">> - (Mandatory Input)
   f) Version:<<Numerical Version>> - (Mandatory Input)
   g) Description:<<Description of the class, method, or property>> - (Optional Input)

Code: -

```java
*CustomAnnotationInfo.java
package annotation;
import java.lang.reflect.Method;
import java.lang.annotation.*;
@Target({ElementType.TYPE,ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
@interface Info
{
int AuthorID();
String Author() default "";
String Supervisor() default "";
String Date();
String Time();
int Version();
String Description() default "";
}

@Info(AuthorID=1,Date="05/11/2021",Time="08:00",Version=15,Description="Custom Annotations Class")
class Demo
{
@Info(AuthorID=1,Author="Mrunal",Date="06/11/2021",Time="09:00",Version=15,Description="Custom Annotations Method")
public void method1()
{
System.out.println("Inside Demo.method1");
}
}
public class CustomAnnotationInfo
{
public static void main(String[] args) throws NoSuchMethodException, SecurityException
{

        Demo obj = new Demo();
        Class c = obj.getClass();
```

```java
*CustomAnnotationInfo.java
}
public class CustomAnnotationInfo
{
public static void main(String[] args) throws NoSuchMethodException, SecurityException
{

        Demo obj = new Demo();
        Class c = obj.getClass();
        Annotation an = c.getAnnotation(Info.class);
        Info i = (Info)an;
        System.out.println("Annotation on Class\n");
        System.out.println("AuthorID:"+i.AuthorID());
        System.out.println("Author:"+i.Author());
        System.out.println("Supervisor:"+i.Supervisor());
        System.out.println("Date:"+i.Date());
        System.out.println("Time:"+i.Time());
        System.out.println("Version:"+i.Version());
        System.out.println("Description:"+i.Description());

        Method m = obj.getClass().getMethod("method1");
        Annotation an1= m.getAnnotation(Info.class);
        Info i1 = (Info)an1;
        System.out.println("\nAnnotation on Method\n");
        System.out.println("AuthorID:"+i1.AuthorID());
        System.out.println("Author:"+i1.Author());
        System.out.println("Supervisor:"+i1.Supervisor());
        System.out.println("Date:"+i1.Date());
        System.out.println("Time:"+i1.Time());
        System.out.println("Version:"+i1.Version());
        System.out.println("Description:"+i1.Description());
    }
}
```

```
Problems  Javadoc  Declaration  Console
<terminated> CustomAnnotationInfo [Java Applicatio
Annotation on Class

AuthorID:1
Author:Mru
Supervisor:
Date:05/11/2021
Time:08:00
Version:15
Description:Custom Annotations Class

Annotation on Method

AuthorID:1
Author:Mrunal
Supervisor:
Date:06/11/2021
Time:09:00
Version:15
Description:Custom Annotations Method
```

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

   E.g.

   Class MyClass{
   @Execute(Sequence=2)
   Public void myMethod1(){
   }
   @Execute(Sequence=1)
   Public void myMethod2(){
   }
   @Execute(Sequence=3)
   Public void myMethod3(){
   }
   }

   Note: The above annotation tells the system that the invocation should be in the order:myMethod2 first, followed by myMethod1 and finally myMethod3

Code: -

CustomAnnotationExecute.java

```java
package annotation;
import java.lang.annotation.ElementType;
import java.lang.annotation.RetentionPolicy;
import java.lang.reflect.Method;
import java.lang.annotation.*;
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
@interface Execute
{
    int Sequence();
}
class MyClass
{
    @Execute(Sequence = 2)
    public void myMethod1() {
        System.out.println("Method 1 should be displayed second");
    }

    @Execute(Sequence = 1)
    public void myMethod2() {
        System.out.println("Method 2 should be displayed first");
    }

    @Execute(Sequence = 3)
    public void myMethod3() {
        System.out.println("Method 3 should be displayed third");
    }
}
public class CustomAnnotationExecute{
```

CustomAnnotationExecute.java

```java
    }

    @Execute(Sequence = 3)
    public void myMethod3() {
        System.out.println("Method 3 should be displayed third");
    }
}
public class CustomAnnotationExecute{
public static void main(String[] args) throws NoSuchMethodException, SecurityException
{
    MyClass obj =new MyClass();

    Method m1=obj.getClass().getMethod("myMethod1");
    Annotation an1 = m1.getAnnotation(Execute.class);
    Execute e1=(Execute)an1;


    Method m2=obj.getClass().getMethod("myMethod2");
    Method m3=obj.getClass().getMethod("myMethod3");


    Annotation an2 = m2.getAnnotation(Execute.class);
    Annotation an3 = m3.getAnnotation(Execute.class);

    Execute e2=(Execute)an2;
    Execute e3=(Execute)an3;
    System.out.println("Method 1 Seq: "+e1.Sequence()+"\nMethod 2 Seq: "+e2.Sequence()+"\nMethod 3 Seq: "+e3.Sequence());

}
}
```

Problems Javadoc Declaration Console Git Staging

```
<terminated> CustomAnnotationExecute [Java Application] C:\Users\MBALKRIS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.0.v20211012-1059\
Method 1 Seq: 2
Method 2 Seq: 1
Method 3 Seq: 3
```