Day 3-Assignment

Bollepalli Sai Sreekanth

1. Write a Program to implement Suppress Annotation.

A.

Code:

```
package sre;

import java.util.ArrayList;

public class suppressWarnings {

    @SuppressWarnings({ "unchecked", "rawtypes" })
    public static void main(String[] args) {
            ArrayList lst=new ArrayList();
            lst.add("1");
            lst.add("2");
            lst.add("3");
            for (Object i:lst) {
                    System.out.println(i);
            }
    }

}
```

Output:

```
1
2
3
```

2. Write a Program to implement Deprecated Annotaion

A.

Code:

```
class h{
    void print1() {
            System.out.println("print 1");
    }
    @Deprecated
    void print2() {
            System.out.println("print 2");
    }
}
public class Main {
    public static void main(String[] args) {
            h a=new h();
            a.print2();}
}
```

Output:

```
Note: ./Main.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
 java -classpath .:target/dependency/* Main
print 2
```

3. Create a custom annotation for the Method

A.

Code:

```
package sre;
```

```
import java.lang.annotation.*;
import java.lang.reflect.*;

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
@interface MyAnnotation{
    int value();
}

class Hello{

    @MyAnnotation(value=50)
    public void sayhello() {
            System.out.println("hello Java Annotations");
    }
}

public class customAnnotation {

    public static void main(String[] args) throws Exception{
            Hello hello=new Hello();
            hello.sayhello();
            Method methodobj=hello.getClass().getMethod("sayhello");
            MyAnnotation
man=methodobj.getAnnotation(MyAnnotation.class);
            System.out.println("value is :"+man.value());


    }
```

}

Output:

```
hello Java Annotations
value is :50
```

4. Create a custom annotation for the Class.

A.

Code:

```
package sre;
import java.lang.annotation.*;
import java.lang.reflect.*;

@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.TYPE})
@interface MyAnnotation{
    int age();
    String name();
    String city();

}
@MyAnnotation(age=21,name="sreekanth",city="Hyderabad")
class Hello{




}
```

```
public class customAnnotation {

    public static void main(String[] args) throws Exception{
            Hello hello=new Hello();
            Class c=hello.getClass();
            System.out.println(c.getName());
            MyAnnotation man=(MyAnnotation)
c.getAnnotation(MyAnnotation.class);

            System.out.println(man.age());
            System.out.println(man.name());
            System.out.println(man.city());

    }

}
```

Output:

```
sre.Hello
   21
sreekanth
Hyderabad
```

5. Write a program which implements two custom annotations.

A.

Code:

```
package sre;
import java.lang.annotation.*;
import java.lang.reflect.*;
```

```
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.TYPE})
@interface MyAnnotation{
    int age();
    String name();
    String city();

}

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
@interface MyyAnnotation{
    int value();
}

@MyAnnotation(age=21,name="sreekanth",city="Hyderabad")
class Hello{




}
class ello{

    @MyyAnnotation(value=50)
    public void sayhello() {
            System.out.println("hello Java Annotations");
    }
}



public class customAnnotation {


    public static void main(String[] args) throws Exception{
            Hello hello=new Hello();
            Class c=hello.getClass();
            System.out.println(c.getName());
            MyAnnotation man=(MyAnnotation)
c.getAnnotation(MyAnnotation.class);


            System.out.println(man.age());
            System.out.println(man.name());
            System.out.println(man.city());


            ello ello=new ello();
            ello.sayhello();
            Method methodobj=ello.getClass().getMethod("sayhello");
            MyyAnnotation
an=methodobj.getAnnotation(MyyAnnotation.class);
            System.out.println("value is :"+an.value());



    }

}
```

Output:

```
sre.Hello
21
sreekanth
Hyderabad
hello Java Annotations
   value is :50
```

6. What is the difference between List<? extends T> and List <? super T>?

A. List<? extends T> represents a list of T or its sub-types such as classes and methods. List<? super T> represents a list of Integer or its super-types of T.

7. Can you pass List<String> to a method which accepts List<Object>?

A. Yes, we can pass List<String > to a method which accepts List<Object>.why because object is a superclass of string. So List<object> can store List<String>.

8. Difference between List<?> and List<Object> in Java?

A. List Is a raw type and it can store any type of list to it but in List<object> can store object into it.

9. What does the string intern() method do in Java?

A. String Interning is a method of storing only one copy of each distinct String Value, which must be immutable. It creates an exact copy of the heap string object in the String Constant Pool.

10. State the difference between String and StringBuffer with Example.

A. String:

=>String is Immutable.

=>String is Thread Safe.

=>Once is String is declared ,It cannot be going to change much.

StringBuilder:

=>String is Mutable. we can change the strings according to use which are declared using StringBuilder . StringBuilder is not ThreadSafe.

Example:

Code:

```
package sre;
public class Q310 {
    public static void add(String s) {
            s=s+"how are you";
    }
    public static void add1(StringBuilder s1) {
            s1.append(",How are you");
    }
    public static void main(String[] args) {
            String s="hi";
            add(s);
            System.out.println("Here String is not Concatenated,so string is
immutable");
            System.out.println("String : "+s);
            StringBuilder s1 = new StringBuilder("Hi");
            add1(s1);
            System.out.println("Here StringBuilder is Concatenated,so
StringBuilder is mutable");
            System.out.println("StringBuilder : "+s1);
    }
}
```

Output:

```
Here String is not Concatenated,so string is immutable
String : hi
Here StringBuilder is Concatenated,so StringBuilder is mutable
StringBuilder : Hi,How are you
```