1401480219

JAVA
ASSIGNMENT-2

Page No.
Date : / /

The different models of security in java are

JavaSE

This model is called code-centric because it allows the specification of code security in a predefined policy file. When a class is loaded in the java run time environment, the class loader following associates the following info with the class.

JAAS

The model extends the java SE code-centric model by introducing a user-centric model where both the nature of the code and the executor are taken into account in security decisions

- JAVA EE

  In this model, secured resources are identified by URL patterns or method names, the container where the applications run enforces authentication and authorisation according to specification in the application deployment descriptor or annotations in the application code.

  - JACC - Permion on JEE resources like URL etc.
  - JPSS - Protects all.

Page No.
Date : / /

2. Similarities

① Interface can not be instantiated, Some way you can not instantiate abstract class.

② That means you can not create obj of interface or abstract class.

Differences

Interfaces	Abstract
• We can use interface keyword to declare interface.	We can use abstract key to declared abs class
• Interface can hold only abstract methods	• Abstract class can hold abstract as well as non abstract methods
• Implemented using implements keywords	Extend using extend keyw

③ Uses of final keyword are to define constants, final

- Used to prevent inheritance
- Prevent overriding
- Method as arguments

④ Super can be used to refer immediate parents class inheritance variables, invoke immediate parent class method and immediate parents class constructor.

Page No.
Date : / /

5 import java.io.*;
import java.lang.*;
import java.util.*;
class shape
{
class 2D
{
public void println("2D")
interface rectangle {
public void print();
}
interface square {
public void print_foo();
}
interface circle extends rectangle, square {
public void print;
}
class child implements circle {
@override public void print_rectangle()
{
System.out.println("rectangle");
}
public void print_foo()
{ system.out.println("circle"); }
}
}
class 3d
{
public void println("3D")

```
interface sphere {
  public void print_sphere();
}
interface cube {
   public void print_cube();
}
interface cube extends sphere {
  @override public void print_sphere()
    {
      System.out.println("sphere");
    }
    public void print_for()
    {
      System.out.println("cube");
    }
}
}
}
}
public class Main {
   public static void main(String[] args)
{
  child c = new child();
    c.print_rectangle();
    c.print_square
    c.print_sphre();
}
}
```

(1)

6. In Derived :: foo()
In Derived :: Bar()
In Base :: foo()
In Base :: Bar()

(ii)

name + nm.member

~~Ans~~