Date:19/11/2014

Name:Venkatesh Kumar B

JAVA

INTERFACE:

- An Interface in Java programming language is defined as an abstract type used to specify the behavior of a class.The interface in Java is *a* mechanism to achieve abstraction
- Interfaces can have abstract methods and variables, but they cannot have method bodies.
- An interface is declared using the interface keyword. All methods in an interface are implicitly public and abstract, and all fields are public, static, and final by default.
- Interface can have only abstract methods before java 8
- Default methods are introduced that can have a body
- The use of default method in interface is for example if the interface is implemented in three classes
- Interfaces cannot contain instance fields or constructors. They are used to define a contract that other classes can implement.

CODE:

```
interface Personal_Details{
        void information();
        default void contact_details() {
        String organization_name = "Steam";
```

```
System.out.println("Organization Name : "+ organization_name);

System.out.println("-----------------------------------");

}

static void permanent_Information() {

System.out.println("Organization Location : United_States");

}

}

public class  Organization_Employee implements Personal_Details {

@Override

public void information () {

String name ="Kumar";

String organization_role = "Developer";

System.out.println("Personal Information");

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

System.out.println("Organization_Role : "+ organization_role);

}

@Override

public void contact_details () {

String organization_name = "Hi Hello  Steam";

System.out.println(organization_name);

System.out.println("-----------------------------------");

}

public static void main(String args[]) {

Organization_Employee  product = new Organization_Employee ();

System.out.println("Default method inside interface");

product. contact_details ();
```

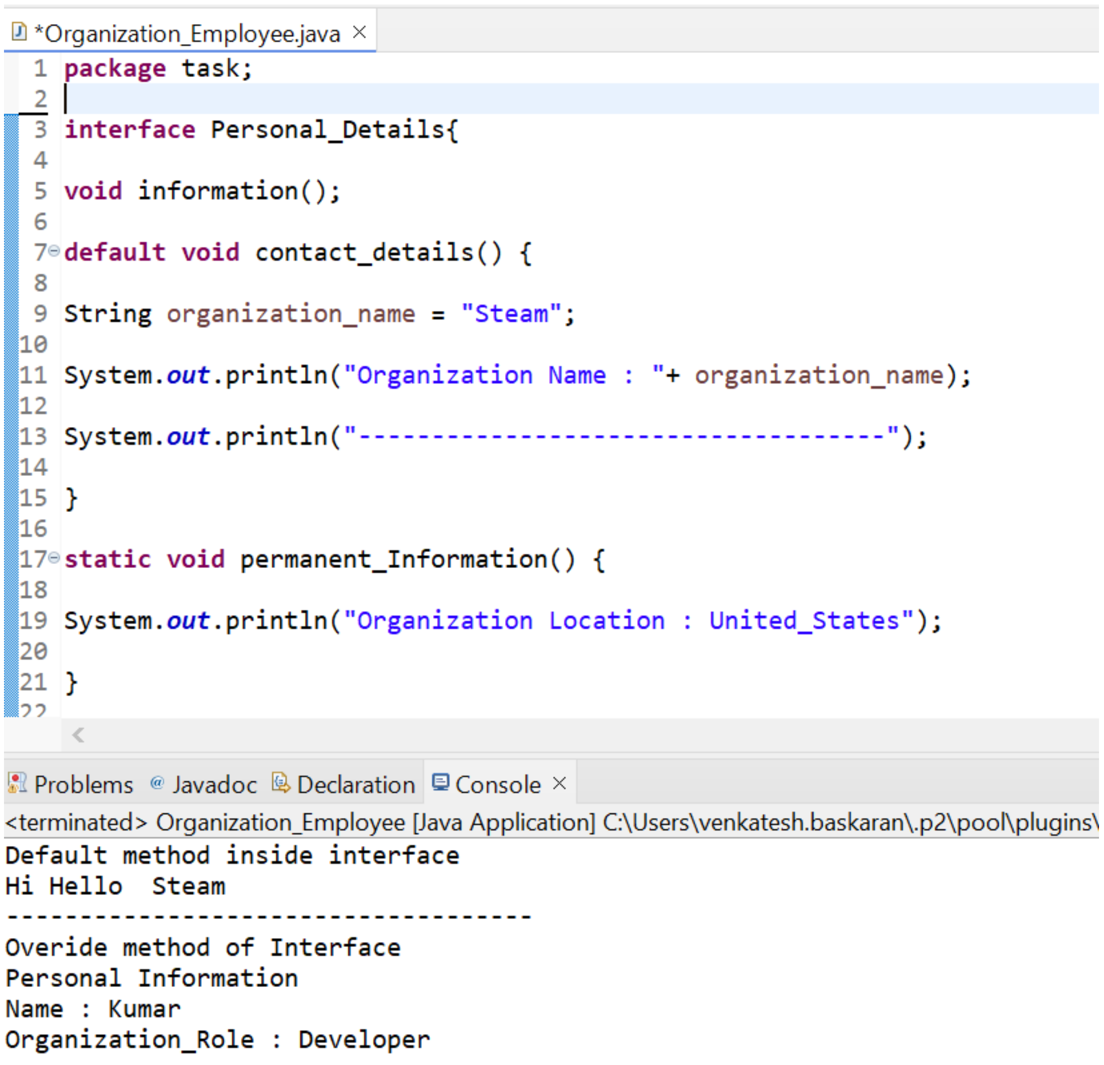
System.out.println("Overide method of Interface");

product. information();

}

}

OUTPUT:



```
package task;

interface Personal_Details{

void information();

default void contact_details() {

String organization_name = "Steam";

System.out.println("Organization Name : "+ organization_name);

System.out.println("-----------------------------------");

}

static void permanent_Information() {

System.out.println("Organization Location : United_States");

}
```

```
<terminated> Organization_Employee [Java Application] C:\Users\venkatesh.baskaran\.p2\pool\plugins\
Default method inside interface
Hi Hello  Steam
-----------------------------------
Overide method of Interface
Personal Information
Name : Kumar
Organization_Role : Developer
```

Abstraction:

- Abstraction in Java is a fundamental concept of object-oriented programming (OOP) that involves hiding the implementation details of a class or an object while exposing only the necessary functionality to the user.This allows the programmer to focus on what an object does rather than how it does it, thereby simplifying the complexity of the code and enhancing modularity.
- An abstract class is a class that cannot be instantiated directly.
- It may contain abstract methods (methods without a body) as well as concrete methods (methods with a body).

Benefits of Abstraction in Java:

- Hides Complexity
- Reusability
- Improved Maintainability
- Loose Coupling

Hides Complexity:

- By abstracting the details, you hide the complexity of how an object works and expose only the relevant operations, making the code easier to use and maintain

Reusability:

- Once abstract methods or classes are defined, subclasses can reuse the code and implement only what is necessary for their specific functionality.

Improved Maintainability:

- Abstraction allows changes in the internal implementation without affecting the code that uses the abstract class or interface

Loose Coupling:

- Abstraction helps achieve loose coupling between components, making the code more flexible and easier to extend.