Experiment – 8

Aim –Adapter design pattern (Structural pattern)

Name – Shivraj Nakum , 21BCP125

Concept -converts the interface of a class into another interface that a client wants . In other words, to provide the interface according to client requirement while using the services of a class with a different interface.

FILES –

Package com.adapter

- Restaurant.java (main method)
- Kitchen (cookFood method)
- Recipes (interface)
- RecipeAdapter (implements interface)

Package com.random

- NewRecipes.java

//

Package com.adapter

Recipes.java

```java
package com.adapter;

public interface Recipes {
    public void cook(String str);
}
```

Kitchen.java

```java
package com.adapter;

public class Kitchen {

    private Recipes r;

    public Recipes getR() {
        return r;
    }

    public void setR(Recipes r) {
        this.r = r;
    }

    public void cookFood(String str){
        r.cook(str);
    }
}
```

RecipeAdapter.java

```
package com.adapter;

import com.random.NewRecipes;

public class RecipeAdapter implements Recipes{

    NewRecipes nr = new NewRecipes();
    @Override
    public void cook(String str) {
        nr.khana(str);
    }
}
```

Restaurant.java

```
package com.adapter;

import com.random.NewRecipes;

public class Restaurant {
    public static void main(String[] args) {

        //NewRecipes nr = new NewRecipes();
        Recipes r = new RecipeAdapter();
        Kitchen kw = new Kitchen();
        kw.setR(r);
        kw.cookFood("I Know to cook recipes");
    }
}
```

<u>Package com.random</u>

NewRecipes.java

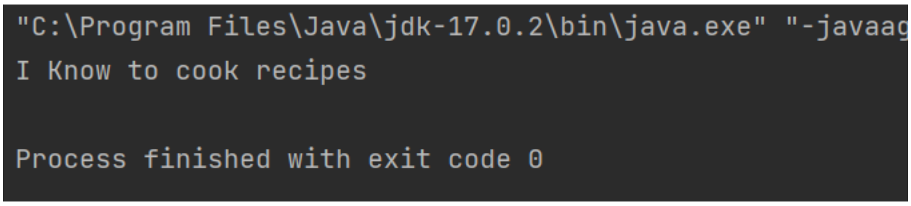
```
package com.random;

public class NewRecipes {

    public void khana(String str){

        System.out.println(str);
    }
}
```

OUTPUT :



```
"C:\Program Files\Java\jdk-17.0.2\bin\java.exe" "-javaag
I Know to cook recipes

Process finished with exit code 0
```