LAB ASSIGNMENT 14

STATE DESIGN PATTERN

State is a behavioral design pattern that lets an object alter its behavior when its internal state changes. It appears as if the object changed its class.

CODE:

- InstrumentOrderState.java

```java
package statedp;

public interface InstrumentOrderState {
    void takeOrder(InstrumentOrder io);

    void prepareInstrument(InstrumentOrder io);

    void deliveredOrder(InstrumentOrder io);
}
```

- InstrumentOrder.java

```java
package statedp;
public class InstrumentOrder {
    private InstrumentOrderState state;

    public InstrumentOrder() {
        state = new OrderingState();
    }

    public void setState(InstrumentOrderState state) {
        this.state = state;
    }

    public void takeOrder() {
        state.takeOrder(this);
    }

    public void prepareInstrument() {
        state.prepareInstrument(this);
    }

    public void deliveredOrder() {
        state.deliveredOrder(this);
    }
}
```

- OrderingState.java

```
package statedp;

public class OrderingState implements InstrumentOrderState {

    @Override
    public void takeOrder(InstrumentOrder io) {
        io.setState(new PreparingState());
        System.out.println("Order for Instrument taken");
    }

    @Override
    public void prepareInstrument(InstrumentOrder io) {
        System.out.println("Instrument can not be built until it is taken");
    }

    @Override
    public void deliveredOrder(InstrumentOrder io) {
        System.out.println("Instrument can not be delivered until it is
built");
    }
}
```

- BuildingState.java

```
package statedp;
public class BuildingState implements InstrumentOrderState {

    @Override
    public void takeOrder(InstrumentOrder io) {
        System.out.println("Order for Instrument has already been taken");
    }

    @Override
    public void prepareInstrument(InstrumentOrder io) {
        io.setState(new BuiltState());
        System.out.println("Instrument is being built");
    }

    @Override
    public void deliveredOrder(InstrumentOrder io) {
        System.out.println("Instrument cannot be delivered until it is
built");
    }
}
```

- BuiltState.java

```java
package statedp;

public class BuiltState implements InstrumentOrderState {

    @Override
    public void takeOrder(InstrumentOrder io) {
        System.out.println("Order for Instrument has already been taken");
    }

    @Override
    public void prepareInstrument(InstrumentOrder io) {
        System.out.println("Instrument has already been built");
    }

    @Override
    public void deliveredOrder(InstrumentOrder io) {
        io.setState(new DeliveredState());
        System.out.println("Instrument is being delivered");
    }
}
```

- DeliveredState.java

```java
package statedp;

public class DeliveredState implements InstrumentOrderState {
    @Override
    public void takeOrder(InstrumentOrder io) {
        System.out.println("Order for Instrument has already been taken");
    }

    @Override
    public void prepareInstrument(InstrumentOrder io) {
        System.out.println("Instrument has already been built");
    }

    @Override
    public void deliveredOrder(InstrumentOrder io) {
        System.out.println("Instrument has already been delivered");
    }
}
```

- Main.java

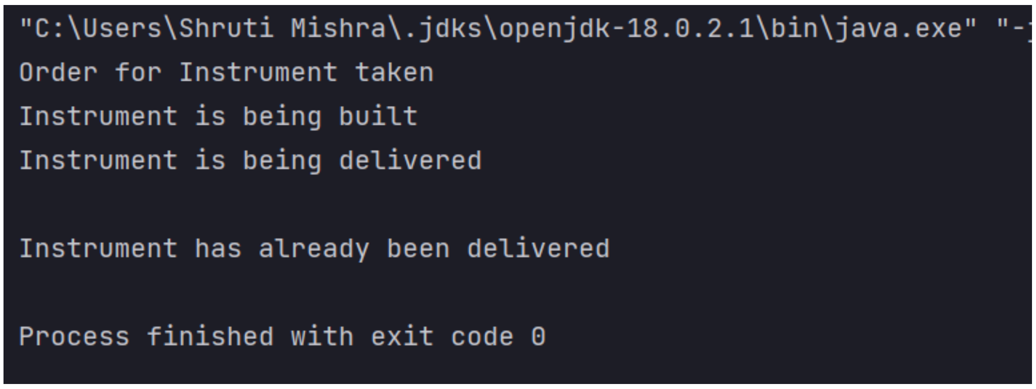
```java
package statedp;

public class Main {
    public static void main(String[] args) {
        InstrumentOrder order = new InstrumentOrder();

        order.takeOrder();
        order.prepareInstrument();
        order.deliveredOrder();

        // Attempt to transition from an invalid state
        System.out.println();
        order.deliveredOrder();
    }
}
```

OUTPUT:



```
"C:\Users\Shruti Mishra\.jdks\openjdk-18.0.2.1\bin\java.exe" "-
Order for Instrument taken
Instrument is being built
Instrument is being delivered

Instrument has already been delivered

Process finished with exit code 0
```