**Youtube.java**

**package** ObserverPattern;

**public** **class** Youtube {

**public** **static** **void** main(String args[])

{

Channel CoolJava = **new** Channel();

Subscriber s1 = **new** Subscriber("Aman");

Subscriber s2 = **new** Subscriber("Bilal");

Subscriber s3 = **new** Subscriber("Chirag");

Subscriber s4 = **new** Subscriber("Disha");

Subscriber s5 = **new** Subscriber("Gayatri");

CoolJava.subscribe(s1);

CoolJava.subscribe(s2);

CoolJava.subscribe(s3);

CoolJava.subscribe(s4);

CoolJava.subscribe(s5);

s1.subscribeChannel(CoolJava);

s2.subscribeChannel(CoolJava);

s3.subscribeChannel(CoolJava);

s4.subscribeChannel(CoolJava);

s5.subscribeChannel(CoolJava);

CoolJava.upload("C'mon lets learn JAVA");

Channel JordarC = **new** Channel();

Subscriber s6 = **new** Subscriber("Amar");

Subscriber s7 = **new** Subscriber("Bansari");

Subscriber s8 = **new** Subscriber("Chintu");

Subscriber s9 = **new** Subscriber("Diksha");

JordarC.subscribe(s6);

JordarC.subscribe(s7);

JordarC.subscribe(s8);

JordarC.subscribe(s9);

JordarC.subscribe(s5);

s6.unsubscribeChannel(JordarC);

JordarC.unsubscribe(s6);

s5.subscribeChannel(JordarC);

s6.subscribeChannel(JordarC);

s7.subscribeChannel(JordarC);

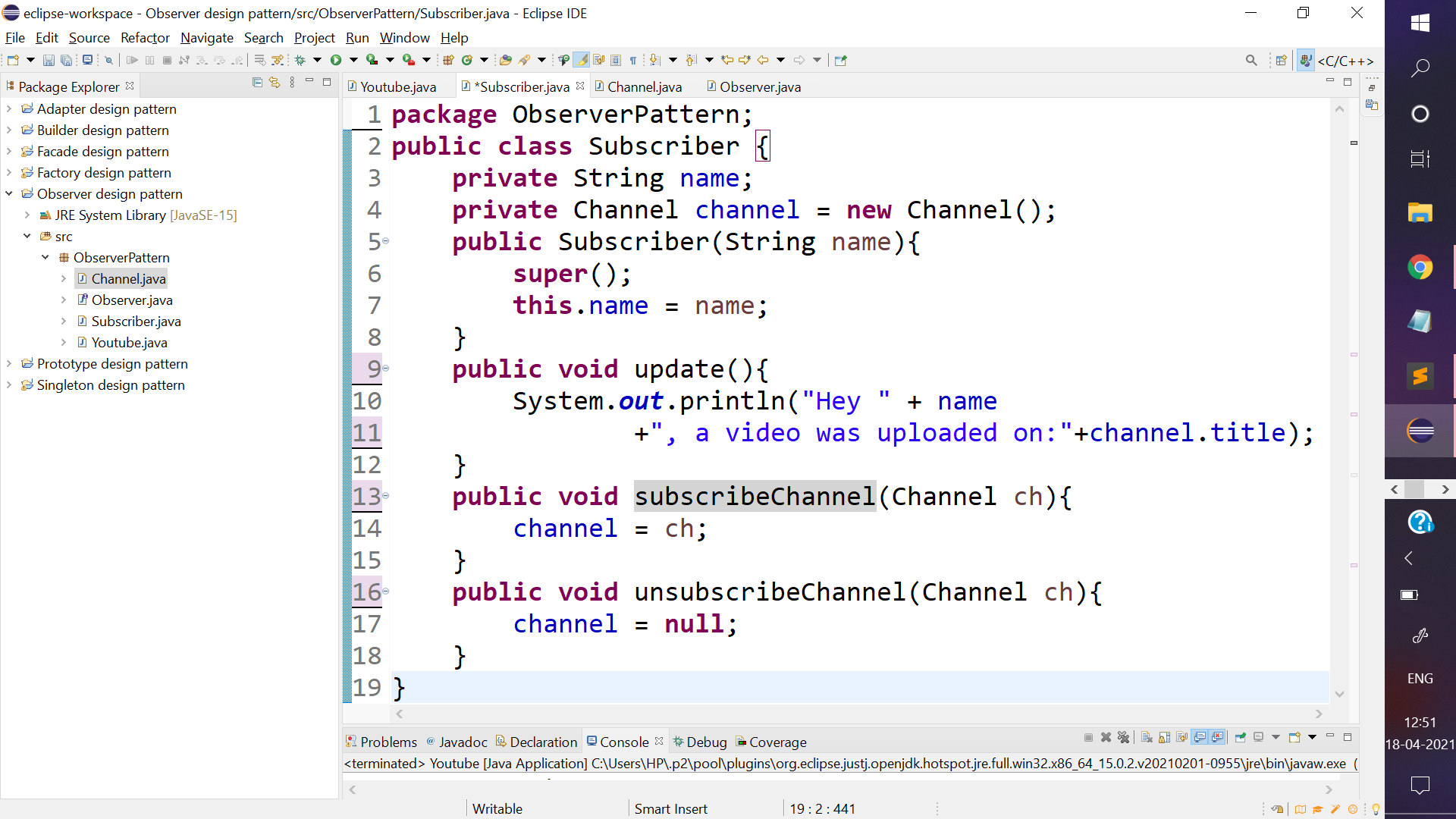
s8.subscribeChannel(JordarC);

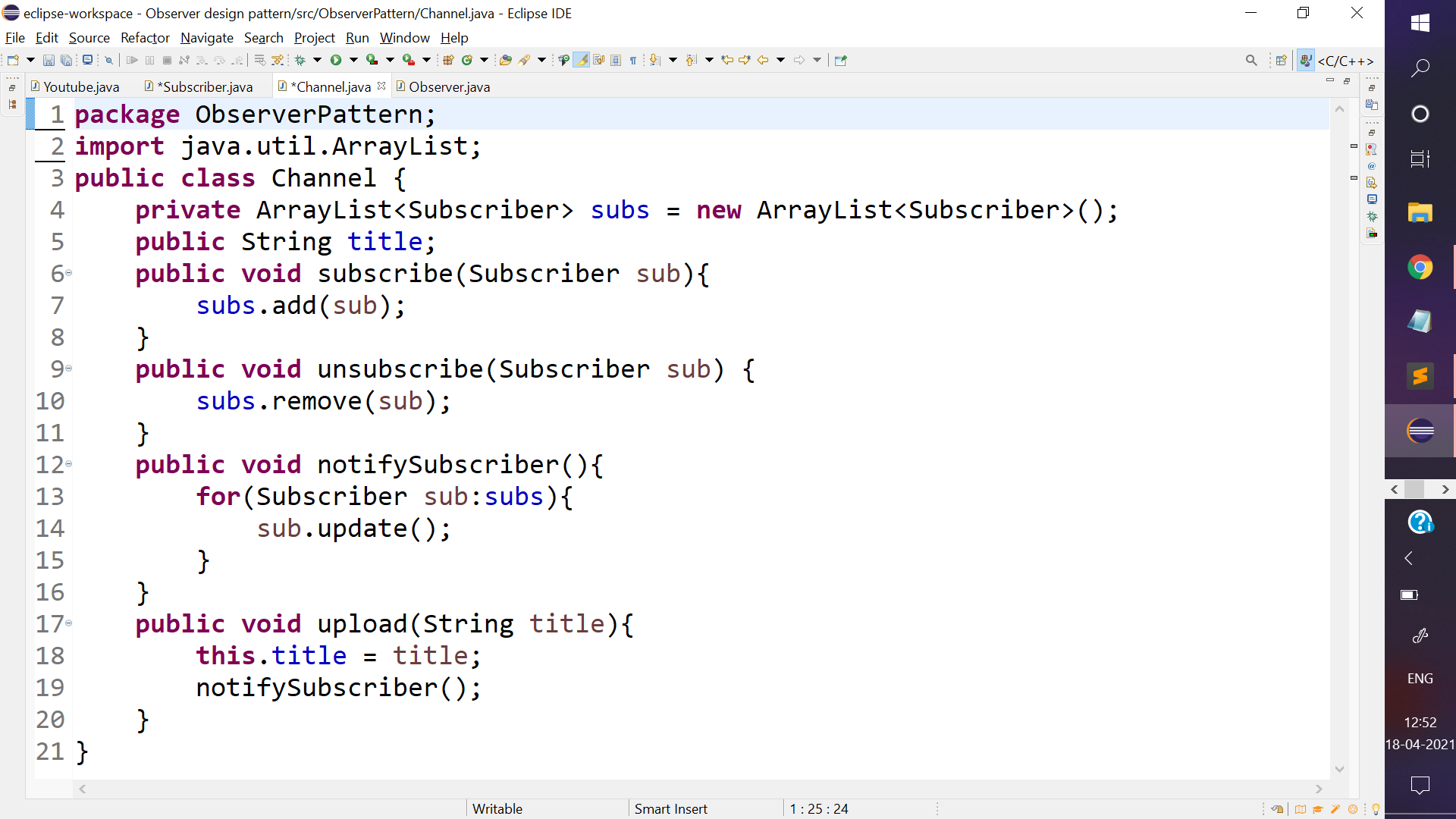
s9.subscribeChannel(JordarC);

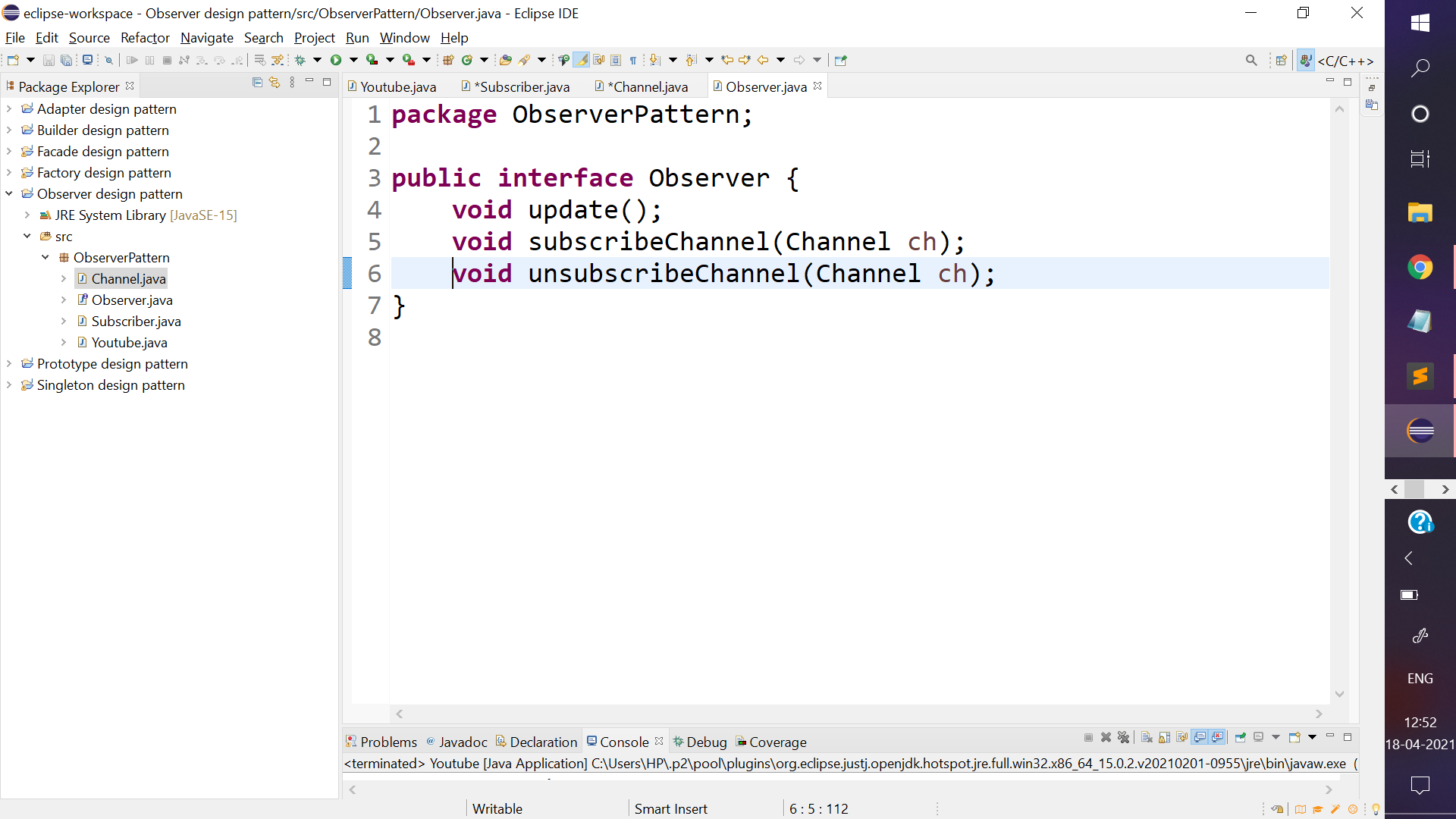
JordarC.upload("C'mon lets learn C/C++ & STL");

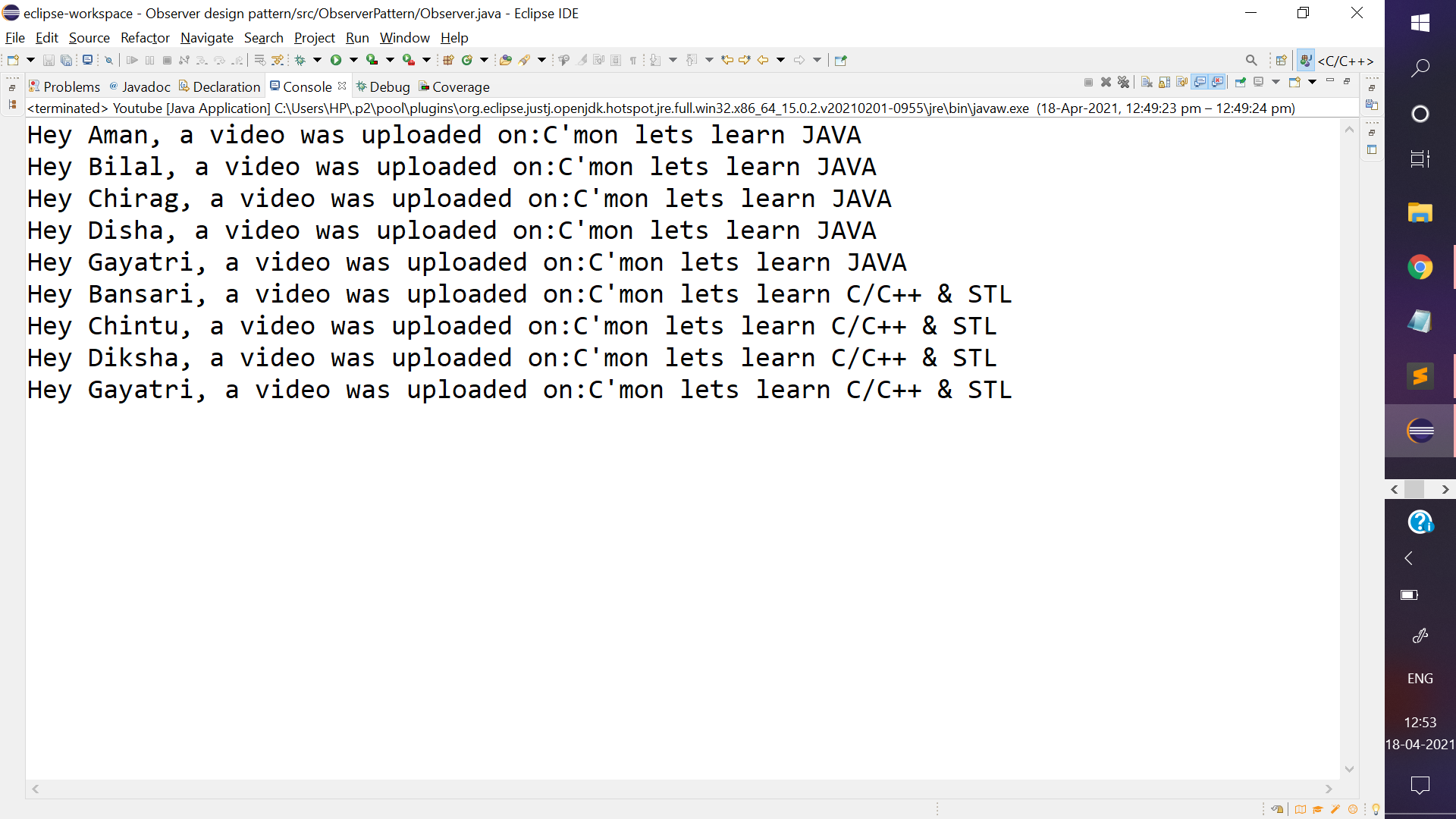
}

}









This is how an Observer design pattern works.

**NOTE:** Look how “Gayatri” has subscribed to both the channels and is receiving notifications for both. On the other hand, “Amar” who had only subscribed one channel but unsubscribed before anything was uploaded on that channel therefore, he does not receive any notifications.