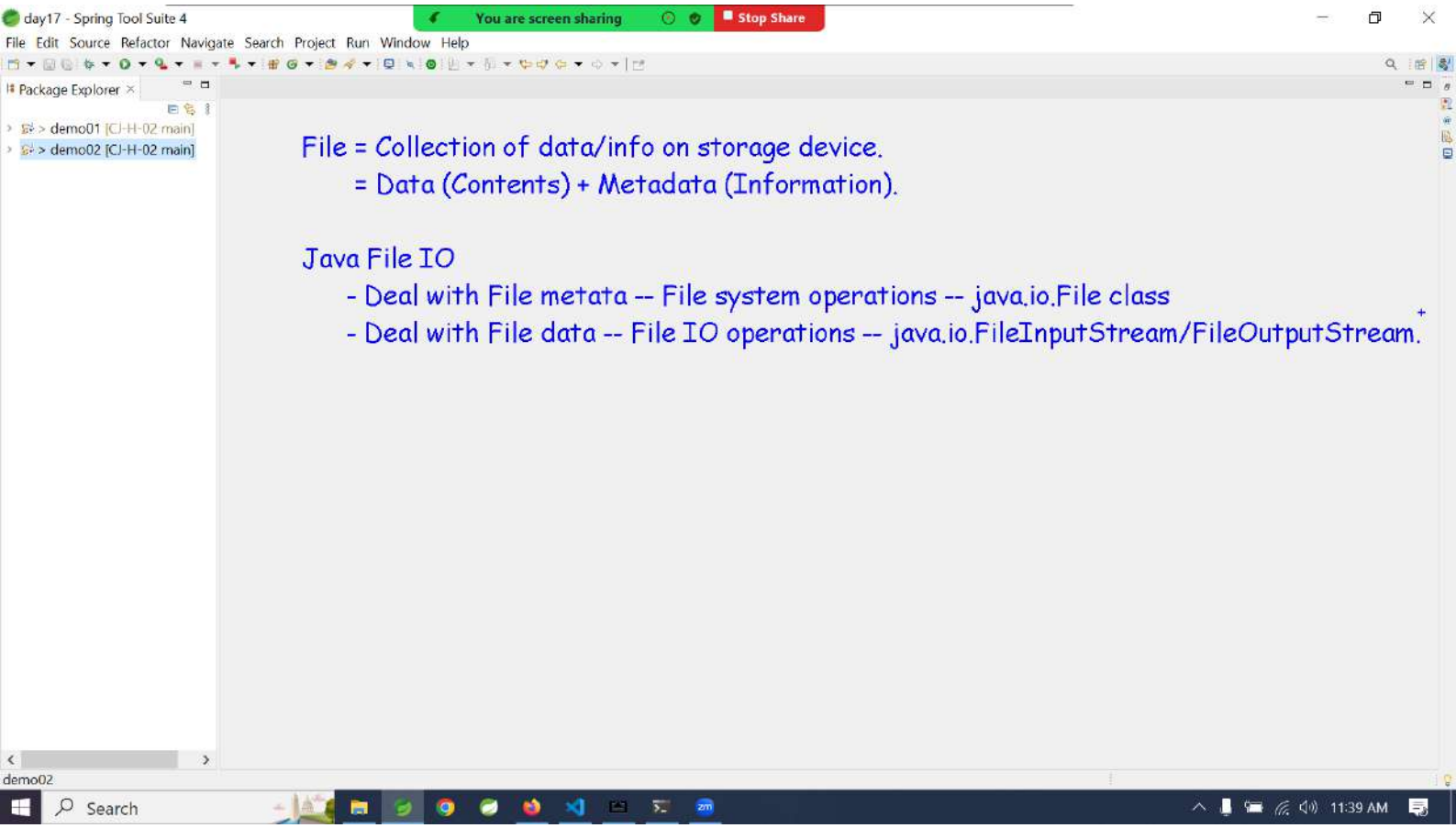
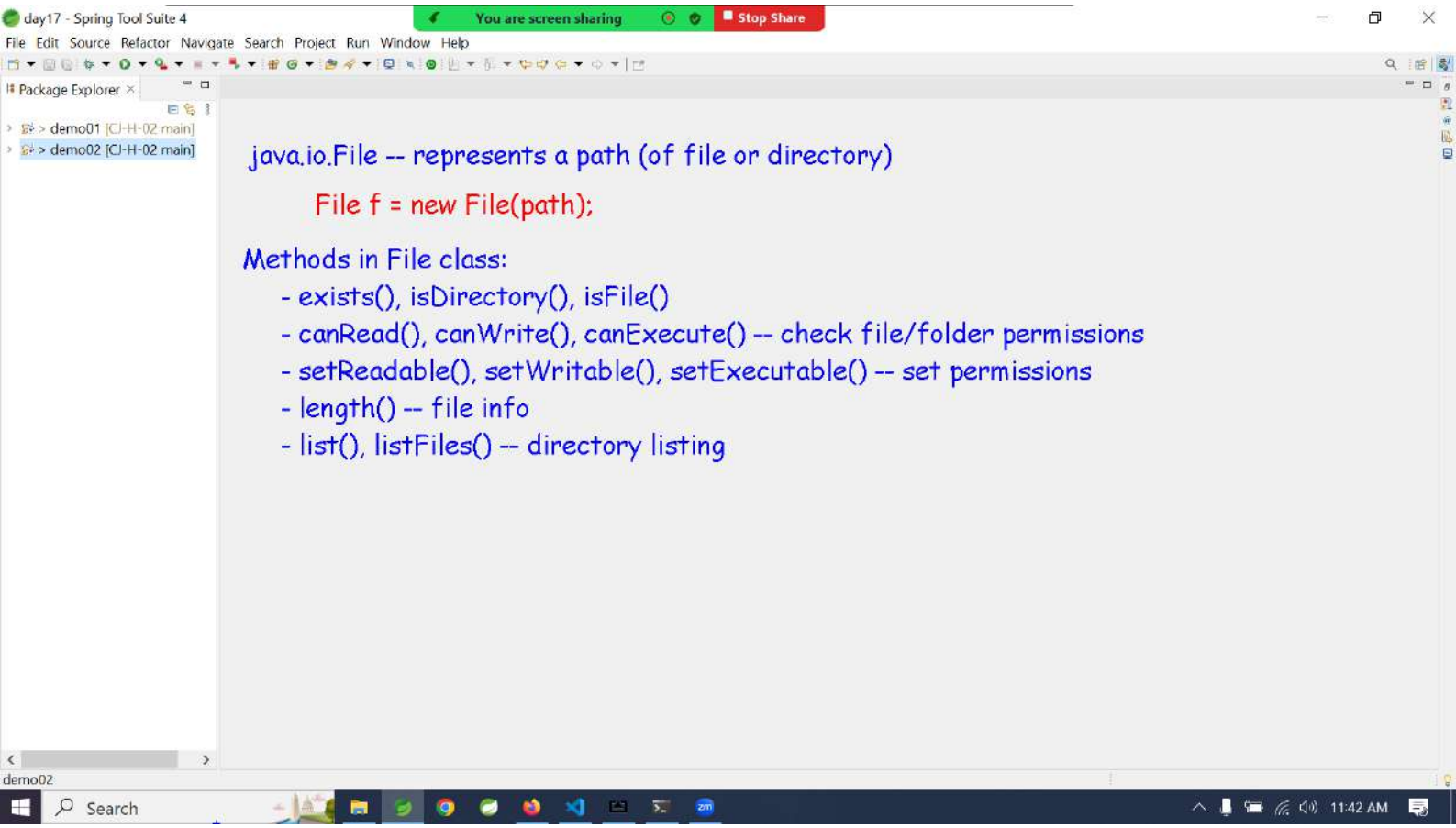
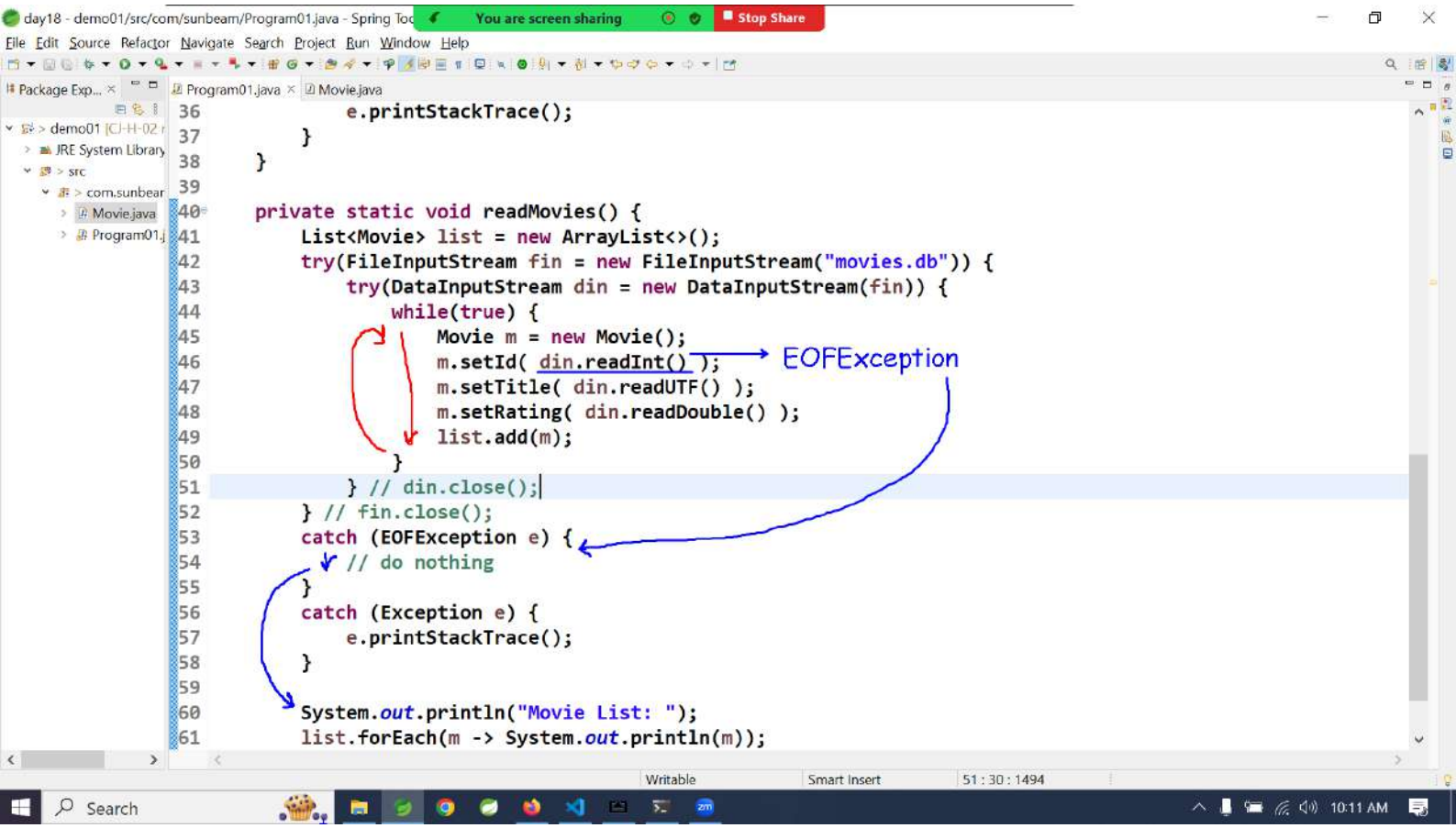

day17 - Spring Tool Suite 4
You are screen sharing
Stop Share
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
> demo01 [CJ-H-02 main]
> demo02 [CJ-H-02 main]
File = Collection of data/info on storage device.
= Data (Contents) + Metadata (Information).
Java File IO
- Deal with File metata -- File system operations -- java.io.File class
- Deal with File data -- File IO operations -- java.io.FileInputStream/FileOutputStream.
demo02
Search
11:39 AM


day17 - Spring Tool Suite 4
You are screen sharing
Stop Share
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
> demo01 [CJ-H-02 main]
> demo02 [CJ-H-02 main]
java.io.File -- represents a path (of file or directory)
File f = new File(path);
Methods in File class:
- exists(), isDirectory(), isFile()
- canRead(), canWrite(), canExecute() -- check file/folder permissions
- setReadable(), setWritable(), setExecutable() -- set permissions
- length() -- file info
- list(), listFiles() -- directory listing
demo02
Search
11:42 AM

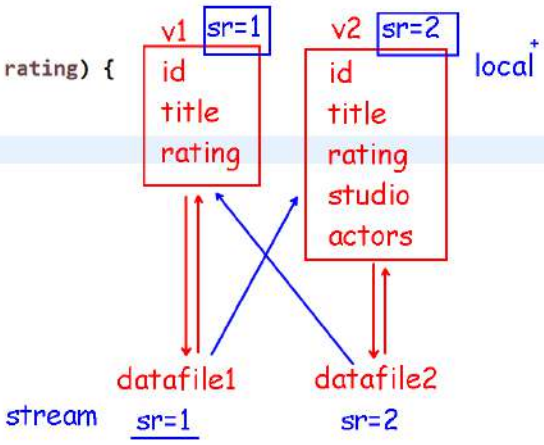

day18 - demo01/src/com/sunbeam/Program01.java - Spring Too
You are screen sharing
Stop Share
File Edit Source Refactor Navigate Search Project Run Window Help
Package Exp...
Program01.java
Movie.java
demo01
JRE System Library
src
com.sunbear
Movie.java
Program01.j
36 e.printStackTrace();
37 }
38 }
39
40 private static void readMovies() {
41 List<Movie> list = new ArrayList<>();
42 try(FileInputStream fin = new FileInputStream("movies.db")) {
43 try(DataInputStream din = new DataInputStream(fin)) {
44 while(true) {
45 Movie m = new Movie();
46 m.setId(din.readInt());
47 m.setTitle(din.readUTF());
48 m.setRating(din.readDouble());
49 list.add(m);
50 }
51 } // din.close();
52 } // fin.close();
53 catch (EOFException e) {
54 // do nothing
55 }
56 catch (Exception e) {
57 e.printStackTrace();
58 }
59
60 System.out.println("Movie List: ");
61 list.forEach(m -> System.out.println(m));
EOFException
Writable
Smart Insert
51 : 30 : 1494
Search
10:11 AM

day18 - demo02/src/com/sunbeam/Movie.java - Spring Tool Suite 4

File Edit Source Refactor Navigate Search Project Run Window Help

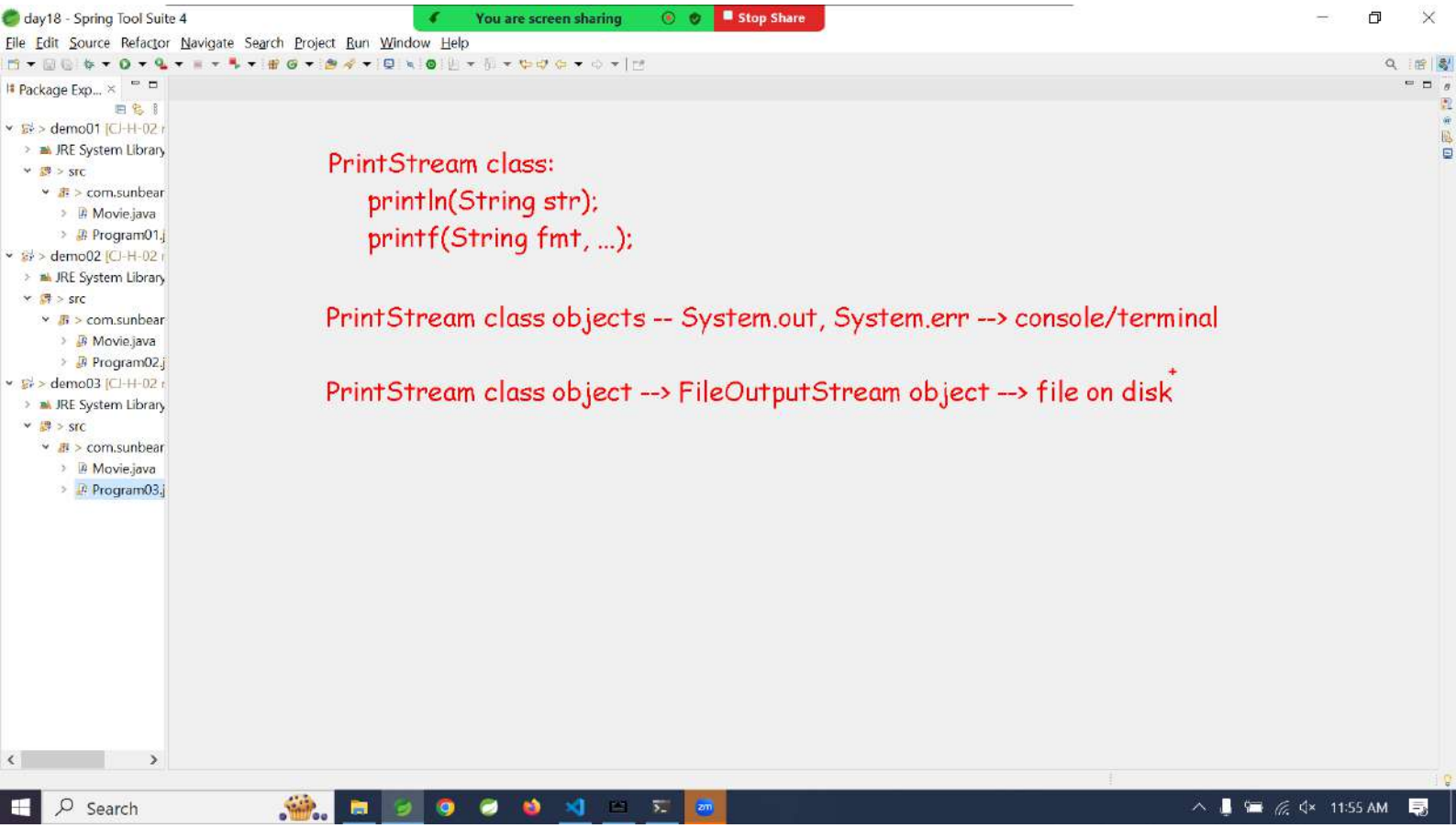
Package Exp... | Program01.java | Movie.java | Program02.java | Movie.java

```
3  import java.io.Serializable;
4
5  public class Movie implements Serializable {
6      private static final long serialVersionUID = 1L; // represents object structure/schema version
7      private int id;
8      private String title;
9      private double rating;
10     transient private String director; // transient fields are not serialized and deserialized
11     static String theatre = "PVR";
12     public Movie() {
13     }
14     public Movie(int id, String title, double rating) {
15         this.id = id;
16         this.title = title;
17         this.rating = rating;
18     }
19     public int getId() {
20         return id;
21     }
22     public void setId(int id) {
23         this.id = id;
24     }
25     public String getTitle() {
26         return title;
27     }
```



Writable | Smart Insert | 17 : 30 : 532

Search | 10:51 AM


day18 - Spring Tool Suite 4
You are screen sharing
Stop Share
File Edit Source Refactor Navigate Search Project Run Window Help
Package Exp...
demo01
JRE System Library
src
com.sunbear
Movie.java
Program01.j
demo02
JRE System Library
src
com.sunbear
Movie.java
Program02.j
demo03
JRE System Library
src
com.sunbear
Movie.java
Program03.j
PrintStream class:
println(String str);
printf(String fmt, ...);
PrintStream class objects -- System.out, System.err --> console/terminal
PrintStream class object --> FileOutputStream object --> file on disk
Search
11:55 AM

Java IO Framework

You are screen sharing | Stop Share

'A' --> 65 (ASCII)

ASCII -- English (1-byte)
UTF-16-- Main lang (2-by)

H L
=hex 2665.
LE 65 26
BE 26 65

Java IO Streams
(flow of data)

Char streams are used to handle char encoding.
Char encoding -- How chars are stored in file/disk.e.g. ASCII, UNICODE (UTF-8, UTF-16, UTF16-BE, UTF16-LE, UTF32, ...). depending on char set to support.

Binary/Byte Streams
(flow of byte)

Character Streams
(flow of chars)

OutputStream
(to write - sink)

```
void write(byte b);*
void write(byte[]);
void flush();
void close();
```

InputStream
(to read - source)

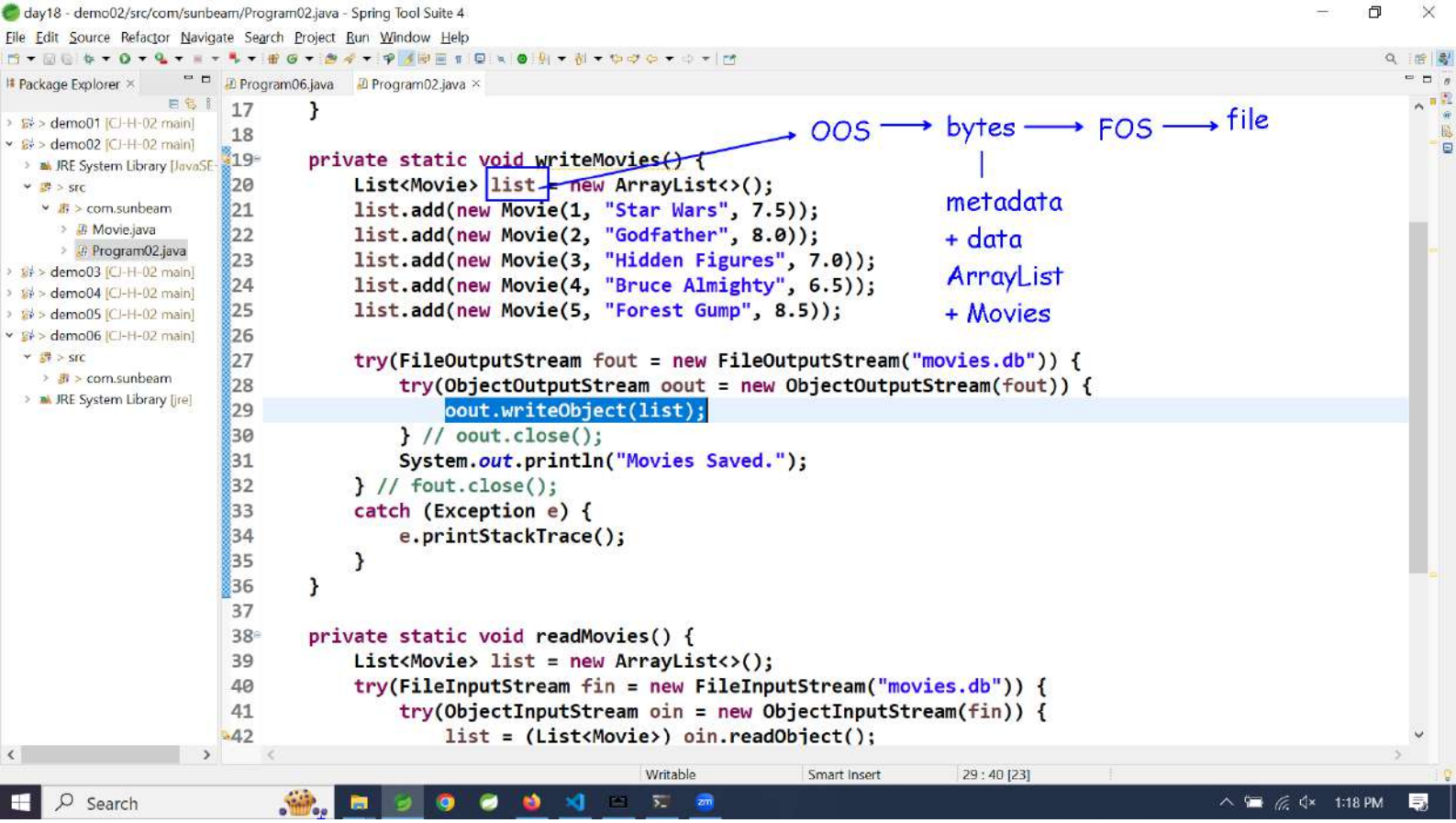
```
int read();*
int read(byte[]);
void close();
```

Writer
(to write - sink)

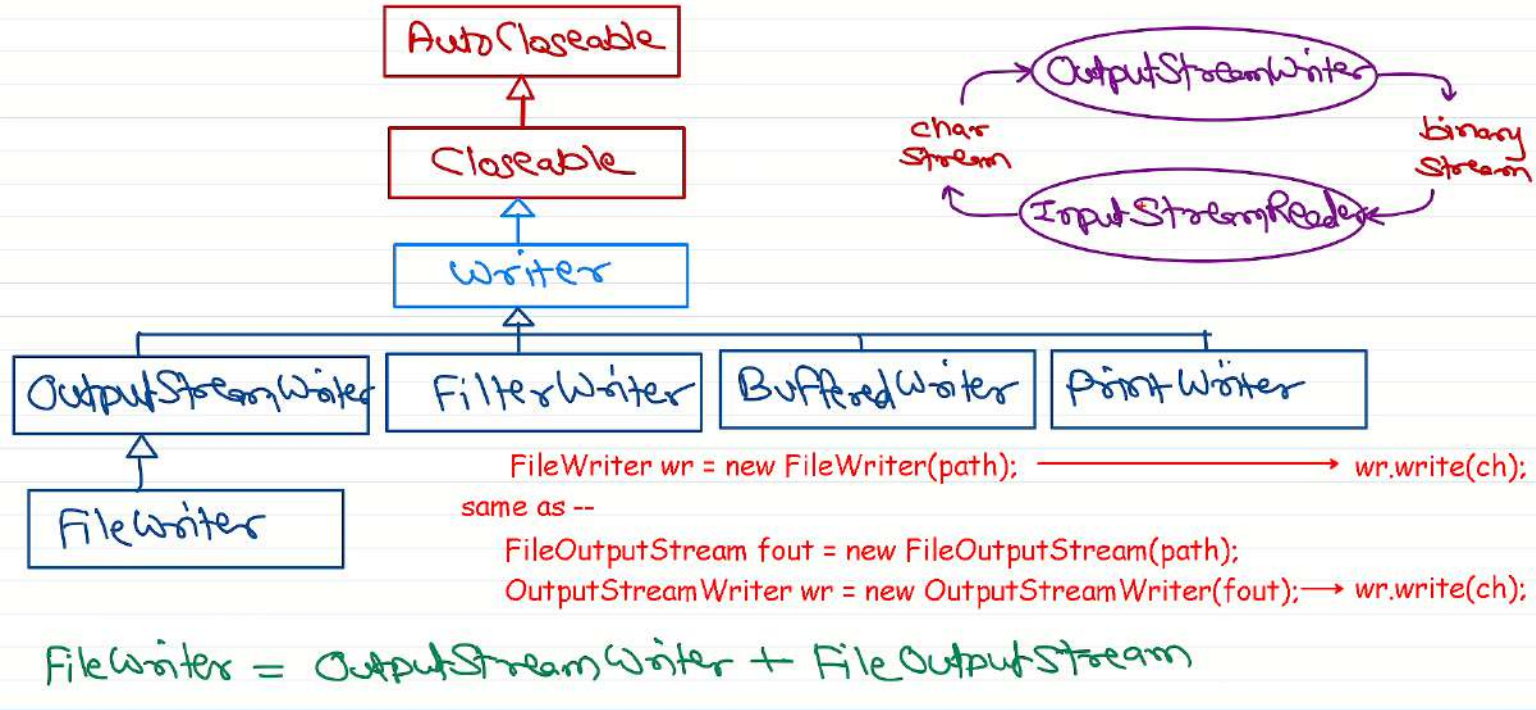
```
void write(char b);*
void write(char[]);
void flush();
void close();
```

Reader
(to read - source)

```
int read();*
int read(char[]);
void close();
```


day18 - demo02/src/com/sunbeam/Program02.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Program06.java
Program02.java
demo01 [CJ-H-02 main]
demo02 [CJ-H-02 main]
JRE System Library [JavaSE
src
com.sunbeam
Movie.java
Program02.java
demo03 [CJ-H-02 main]
demo04 [CJ-H-02 main]
demo05 [CJ-H-02 main]
demo06 [CJ-H-02 main]
src
com.sunbeam
JRE System Library [jre]
17 }
18
19 private static void writeMovies() {
20 List<Movie> list = new ArrayList<>();
21 list.add(new Movie(1, "Star Wars", 7.5));
22 list.add(new Movie(2, "Godfather", 8.0));
23 list.add(new Movie(3, "Hidden Figures", 7.0));
24 list.add(new Movie(4, "Bruce Almighty", 6.5));
25 list.add(new Movie(5, "Forest Gump", 8.5));
26
27 try(FileOutputStream fout = new FileOutputStream("movies.db")) {
28 try(ObjectOutputStream oout = new ObjectOutputStream(fout)) {
29 oout.writeObject(list);
30 } // oout.close();
31 System.out.println("Movies Saved.");
32 } // fout.close();
33 catch (Exception e) {
34 e.printStackTrace();
35 }
36 }
37
38 private static void readMovies() {
39 List<Movie> list = new ArrayList<>();
40 try(FileInputStream fin = new FileInputStream("movies.db")) {
41 try(ObjectInputStream oin = new ObjectInputStream(fin)) {
42 list = (List<Movie>) oin.readObject();
OOS → bytes → FOS → file
metadata
+ data
ArrayList
+ Movies
Writable
Smart Insert
29 : 40 [23]
Search
1:18 PM

Writer hierarchy

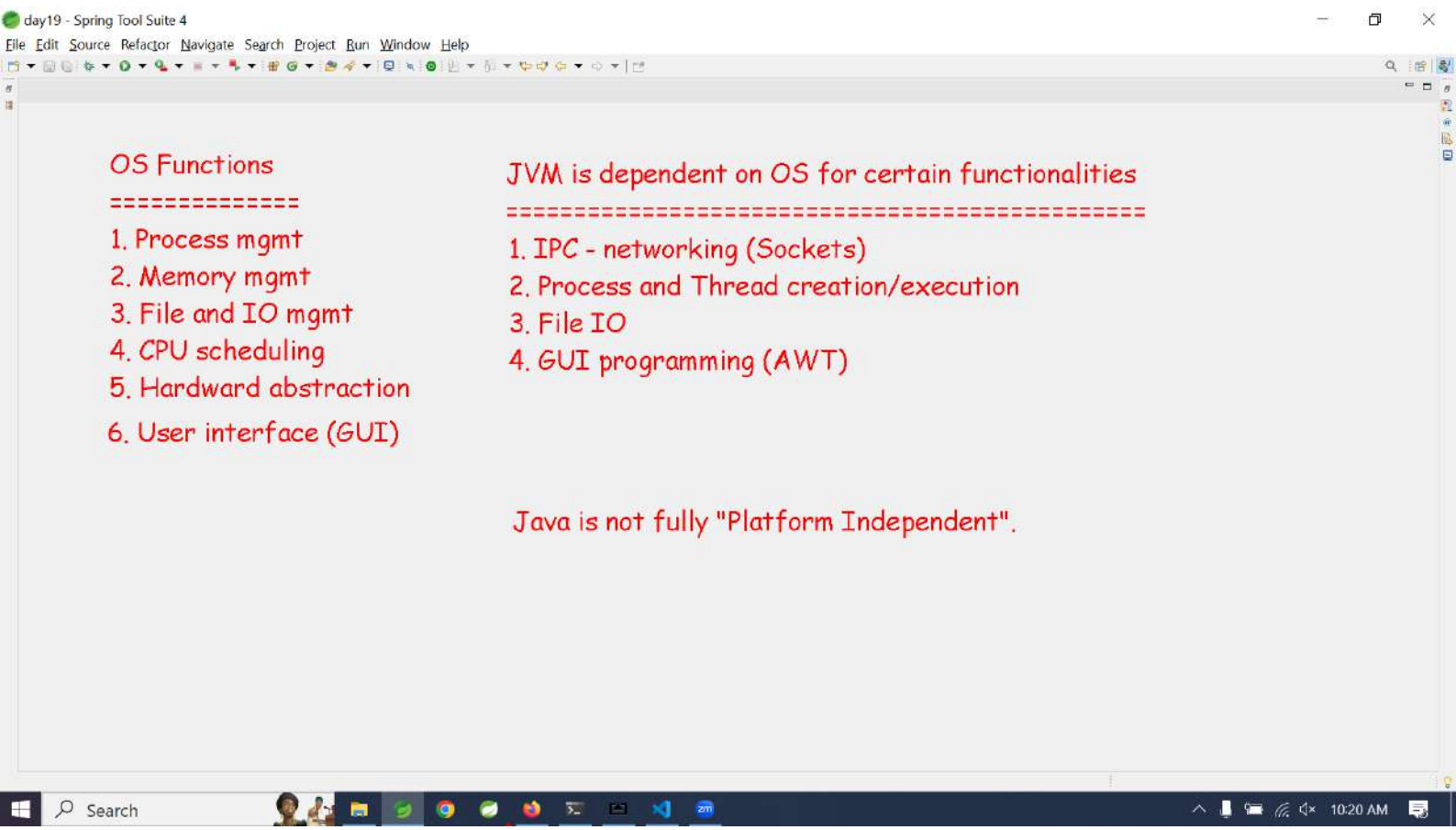


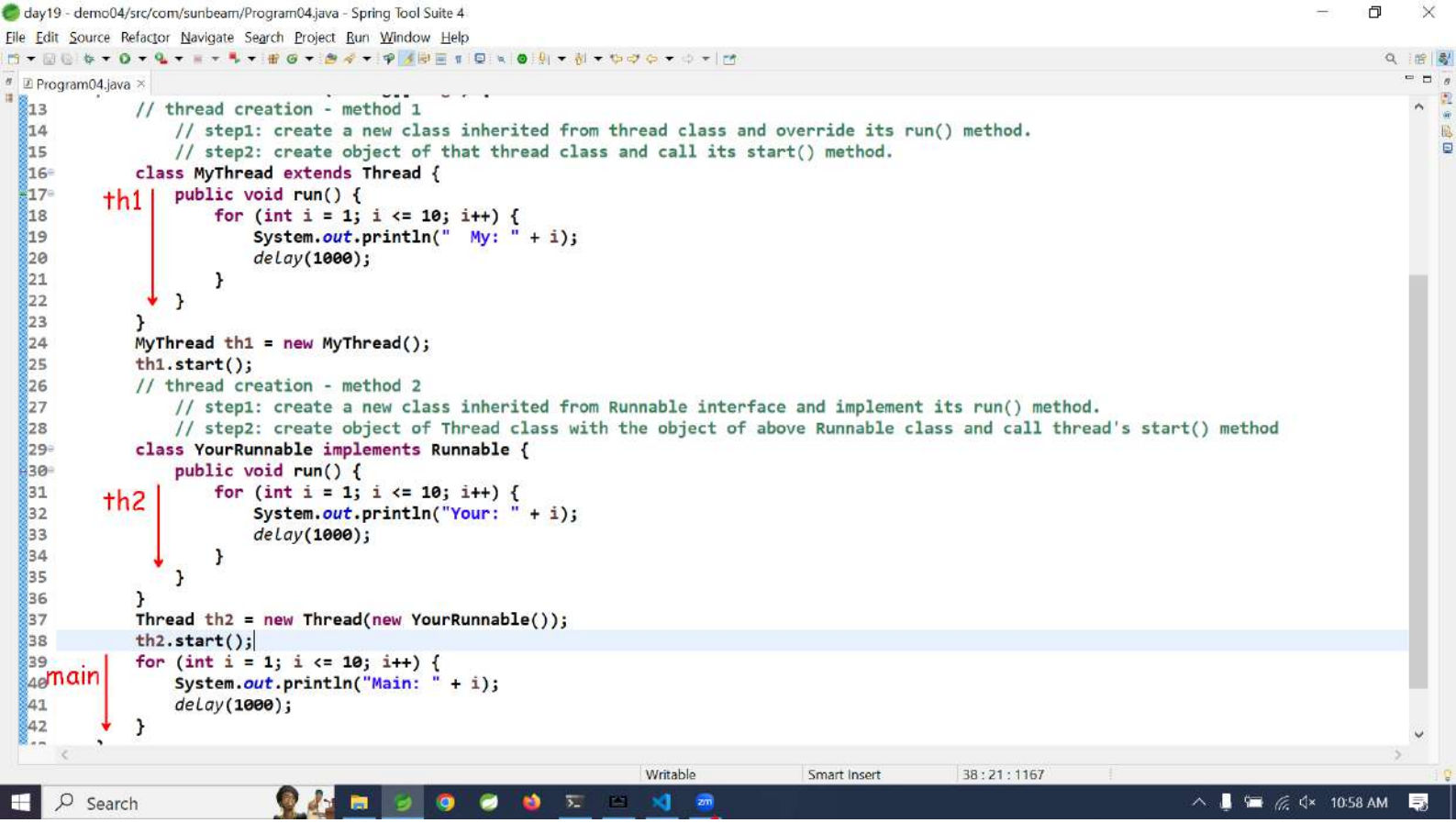

```
FileWriter wr = new FileWriter(path);  ------>  wr.write(ch);
```

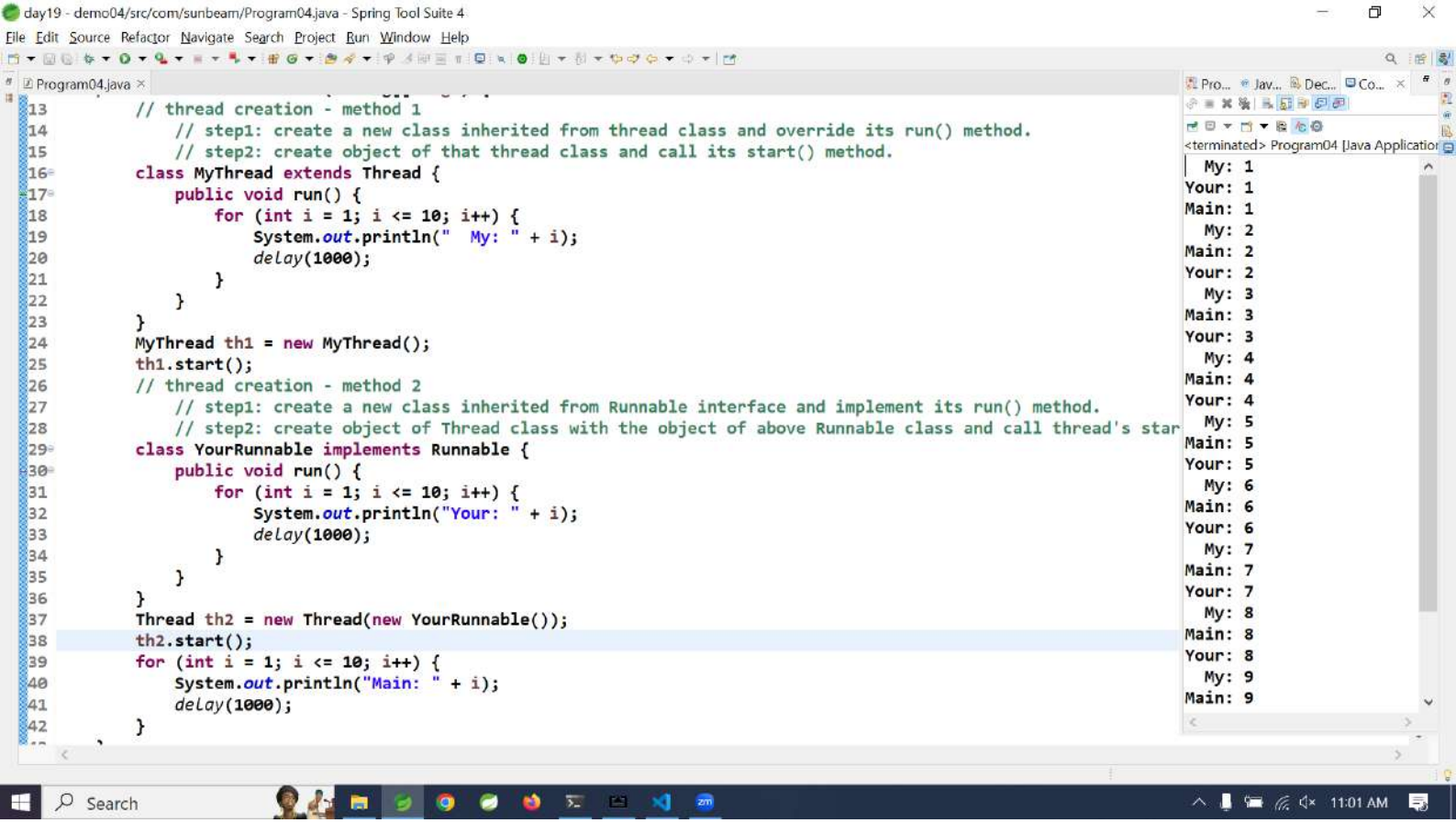
same as --

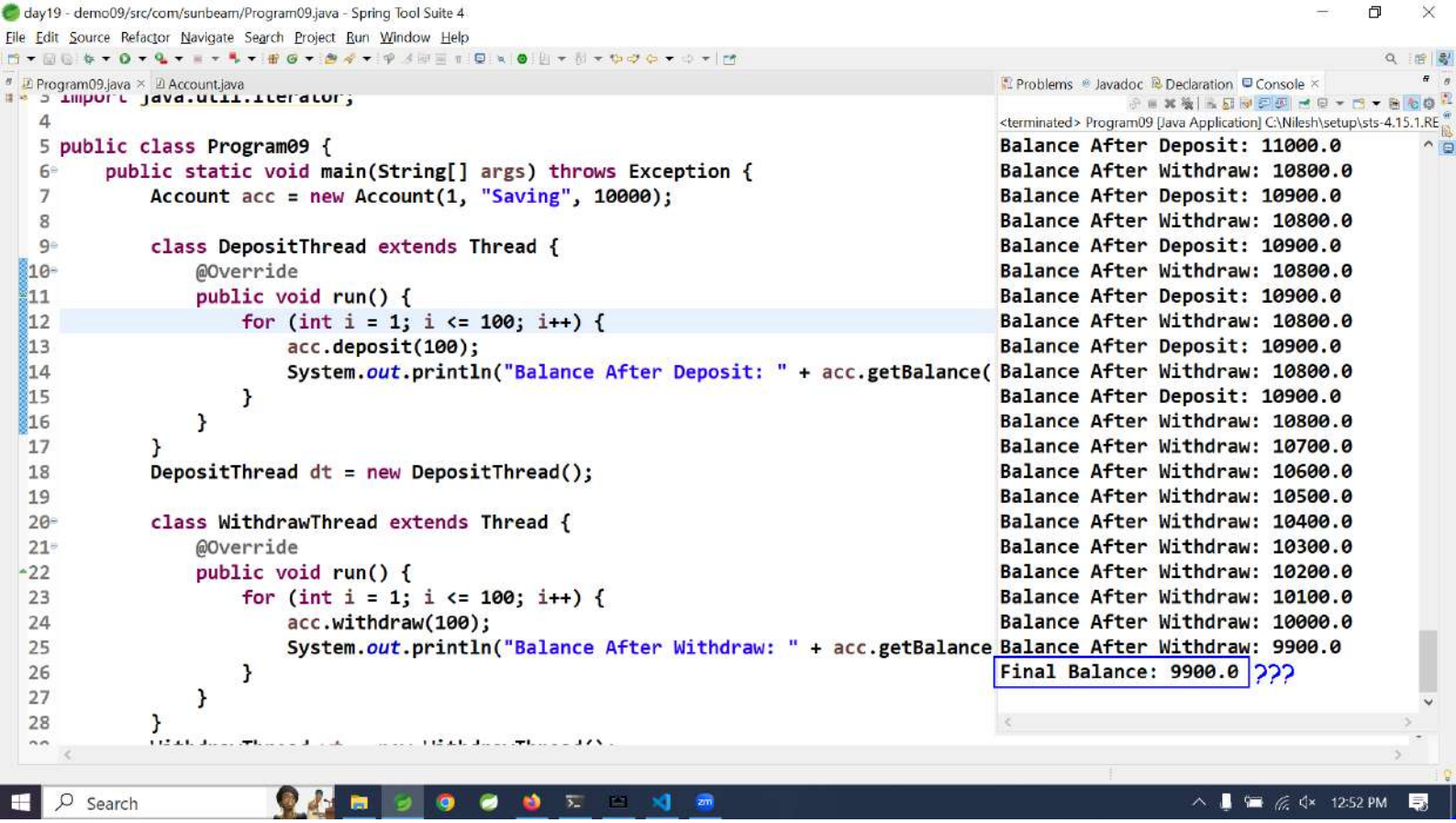
```
FileOutputStream fout = new FileOutputStream(path);
OutputStreamWriter wr = new OutputStreamWriter(fout);--> wr.write(ch);
```

FileWriter = OutputStreamWriter + FileOutputStream


day19 - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
OS Functions
==========
1. Process mgmt
2. Memory mgmt
3. File and IO mgmt
4. CPU scheduling
5. Hardward abstraction
6. User interface (GUI)
JVM is dependent on OS for certain functionalities
==========
1. IPC - networking (Sockets)
2. Process and Thread creation/execution
3. File IO
4. GUI programming (AWT)
Java is not fully "Platform Independent".
Search
10:20 AM


day19 - demo04/src/com/sunbeam/Program04.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Program04.java
// thread creation - method 1
// step1: create a new class inherited from thread class and override its run() method.
// step2: create object of that thread class and call its start() method.
class MyThread extends Thread {
th1
public void run() {
for (int i = 1; i <= 10; i++) {
System.out.println(" My: " + i);
delay(1000);
}
}
}
MyThread th1 = new MyThread();
th1.start();
// thread creation - method 2
// step1: create a new class inherited from Runnable interface and implement its run() method.
// step2: create object of Thread class with the object of above Runnable class and call thread's start() method
class YourRunnable implements Runnable {
public void run() {
th2
for (int i = 1; i <= 10; i++) {
System.out.println("Your: " + i);
delay(1000);
}
}
}
Thread th2 = new Thread(new YourRunnable());
th2.start();
for (int i = 1; i <= 10; i++) {
main
System.out.println("Main: " + i);
delay(1000);
}
Writable
Smart Insert
38 : 21 : 1167
Search
10:58 AM


day19 - demo04/src/com/sunbeam/Program04.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Program04.java
// thread creation - method 1
// step1: create a new class inherited from thread class and override its run() method.
// step2: create object of that thread class and call its start() method.
class MyThread extends Thread {
public void run() {
for (int i = 1; i <= 10; i++) {
System.out.println(" My: " + i);
delay(1000);
}
}
}
MyThread th1 = new MyThread();
th1.start();
// thread creation - method 2
// step1: create a new class inherited from Runnable interface and implement its run() method.
// step2: create object of Thread class with the object of above Runnable class and call thread's star
class YourRunnable implements Runnable {
public void run() {
for (int i = 1; i <= 10; i++) {
System.out.println("Your: " + i);
delay(1000);
}
}
}
Thread th2 = new Thread(new YourRunnable());
th2.start();
for (int i = 1; i <= 10; i++) {
System.out.println("Main: " + i);
delay(1000);
}
<terminated> Program04 [Java Application
My: 1
Your: 1
Main: 1
My: 2
Main: 2
Your: 2
My: 3
Main: 3
Your: 3
My: 4
Main: 4
Your: 4
My: 5
Main: 5
Your: 5
My: 6
Main: 6
Your: 6
My: 7
Main: 7
Your: 7
My: 8
Main: 8
Your: 8
My: 9
Main: 9
Search
11:01 AM


day19 - demo09/src/com/sunbeam/Program09.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Program09.java
Account.java
Problems Javadoc Declaration Console
<terminated> Program09 [Java Application] C:\Nilesh\setup\sts-4.15.1.RE
4
5 public class Program09 {
6 public static void main(String[] args) throws Exception {
7 Account acc = new Account(1, "Saving", 10000);
8
9 class DepositThread extends Thread {
10 @Override
11 public void run() {
12 for (int i = 1; i <= 100; i++) {
13 acc.deposit(100);
14 System.out.println("Balance After Deposit: " + acc.getBalance(
15 }
16 }
17 }
18 DepositThread dt = new DepositThread();
19
20 class WithdrawThread extends Thread {
21 @Override
22 public void run() {
23 for (int i = 1; i <= 100; i++) {
24 acc.withdraw(100);
25 System.out.println("Balance After Withdraw: " + acc.getBalance
26 }
27 }
28 }
Balance After Deposit: 11000.0
Balance After Withdraw: 10800.0
Balance After Deposit: 10900.0
Balance After Withdraw: 10800.0
Balance After Deposit: 10900.0
Balance After Withdraw: 10800.0
Balance After Deposit: 10900.0
Balance After Withdraw: 10800.0
Balance After Deposit: 10900.0
Balance After Withdraw: 10800.0
Balance After Deposit: 10900.0
Balance After Withdraw: 10800.0
Balance After Withdraw: 10700.0
Balance After Withdraw: 10600.0
Balance After Withdraw: 10500.0
Balance After Withdraw: 10400.0
Balance After Withdraw: 10300.0
Balance After Withdraw: 10200.0
Balance After Withdraw: 10100.0
Balance After Withdraw: 10000.0
Balance After Withdraw: 9900.0
Final Balance: 9900.0
???
Search
12:52 PM