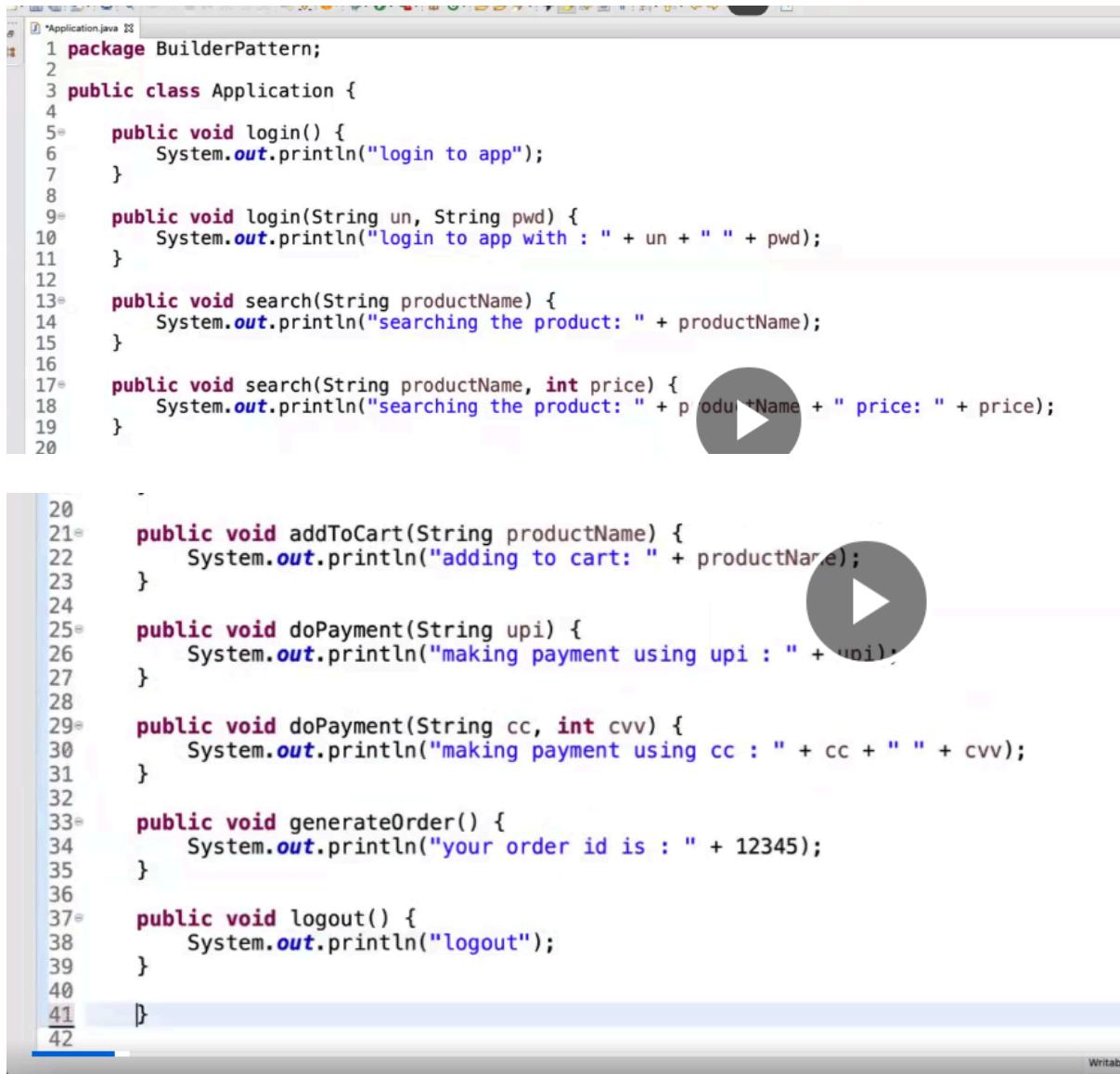


Builder pattern-

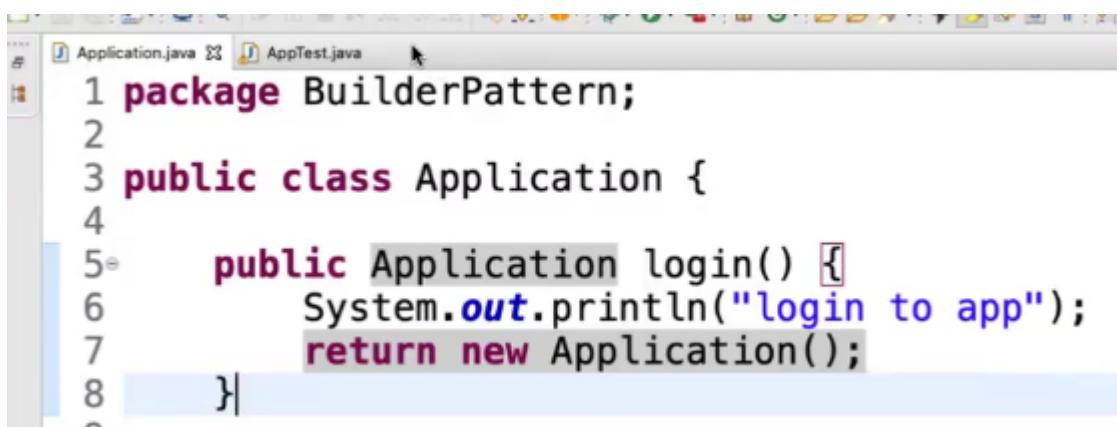


```

1 package BuilderPattern;
2
3 public class Application {
4
5     public void login() {
6         System.out.println("login to app");
7     }
8
9     public void login(String un, String pwd) {
10        System.out.println("login to app with : " + un + " " + pwd);
11    }
12
13    public void search(String productName) {
14        System.out.println("searching the product: " + productName);
15    }
16
17    public void search(String productName, int price) {
18        System.out.println("searching the product: " + productName + " price: " + price);
19    }
20
21    public void addToCart(String productName) {
22        System.out.println("adding to cart: " + productName);
23    }
24
25    public void doPayment(String upi) {
26        System.out.println("making payment using upi : " + upi);
27    }
28
29    public void doPayment(String cc, int cvv) {
30        System.out.println("making payment using cc : " + cc + " " + cvv);
31    }
32
33    public void generateOrder() {
34        System.out.println("your order id is : " + 12345);
35    }
36
37    public void logout() {
38        System.out.println("logout");
39    }
40
41 }
42

```

Login returning the object of application class-

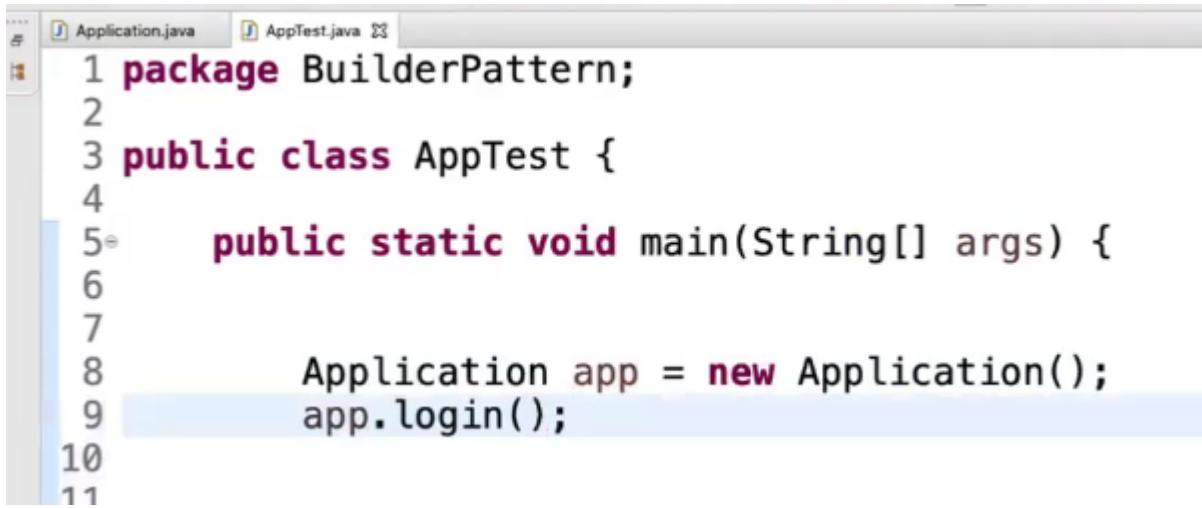


```

1 package BuilderPattern;
2
3 public class Application {
4
5     public Application login() {
6         System.out.println("login to app");
7         return new Application();
8     }
9

```

Call login-



```

1 package BuilderPattern;
2
3 public class Application {
4
5     public Application login() {
6         System.out.println("login to app");
7         return this;
8     }
9 }
10
11
12 public class AppTest {
13
14     public static void main(String[] args) {
15
16         Application app = new Application();
17         app.login();
18     }
19 }

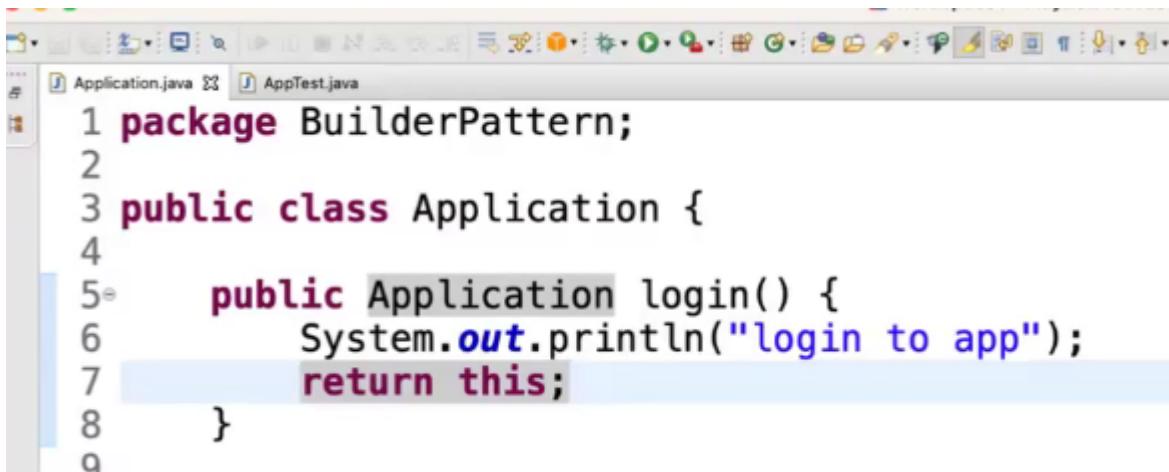
```

Problems-

Two objects created – one in application class when we wrote “return new application” and other in the above class when we call the login. Memory wastage.

In application class, there is no reference to the object, so garbage collector can destroy it anytime.

This will solve the problem-



```

1 package BuilderPattern;
2
3 public class Application {
4
5     public Application login() {
6         System.out.println("login to app");
7         return this;
8     }
9 }
10
11
12 public class AppTest {
13
14     public static void main(String[] args) {
15
16         Application app = new Application();
17         app.login();
18     }
19 }

```

Return “this;” means return the current class object if any existing in memory.

See memory wastage when we write
“return new Application;”-



```
1 package BuilderPattern;
2
3 public class Application {
4
5     public Application login() {
6         System.out.println("login to app");
7         return new Application();
8     }
9
10    public void login(String un, String pwd) {
11        System.out.println("login to app with : " + un + " " + pwd);
12    }
13
14    public void search(String productName) {
15        System.out.println("searching the product: " + productName);
16    }
17
18    public void search(String productName, int price) {
19        System.out.println("searching the product: " + productName + " price: " + price);
20    }
21
22    public void addToCart(String productName) {
23        System.out.println("adding to cart: " + productName);
24    }
25
26    public void doPayment(String upi) {
27        System.out.println("processing payment via " + upi);
28    }
29}
```

Every method returns this-

```

4
5  public Application login() {
6      System.out.println("login to app");
7      return this;
8  }
9
10 public Application login(String un, String pwd) {
11     System.out.println("login to app with : " + un + " " + pwd);
12     return this;
13 }
14
15 public Application search(String productName) {
16     System.out.println("searching the product: " + productName);
17     return this;
18 }
19
20 public Application search(String productName, int price) {
21     System.out.println("searching the product: " + productName + " price: " + price);
22     return this;
23 }
24
25 public Application addToCart(String productName) {
26     System.out.println("adding to cart: " + productName);
27     return this;
28 }
29
30 public Application doPayment(String upi) {
31     System.out.println("making payment using upi : " + upi);
32     return this;
33 }
34
35 public Application doPayment(String cc, int cvv) {
36     System.out.println("making payment using cc : " + cc + " " + cvv);
37     return this;
38 }
39
40 public Application generateOrder() {
41     System.out.println("your order id is : " + 12345);
42     return this;
43 }
44

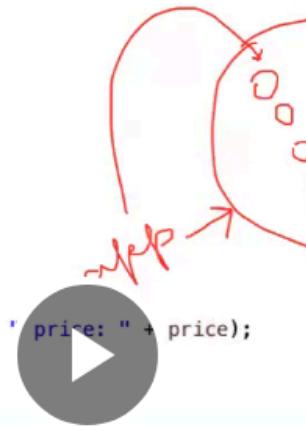
```

```

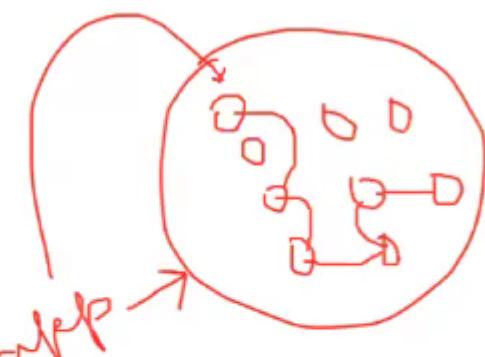
44
45 public void logout() {
46     System.out.println("logout");
47     return this;
48 }
49
50 }

```

Builder pattern-

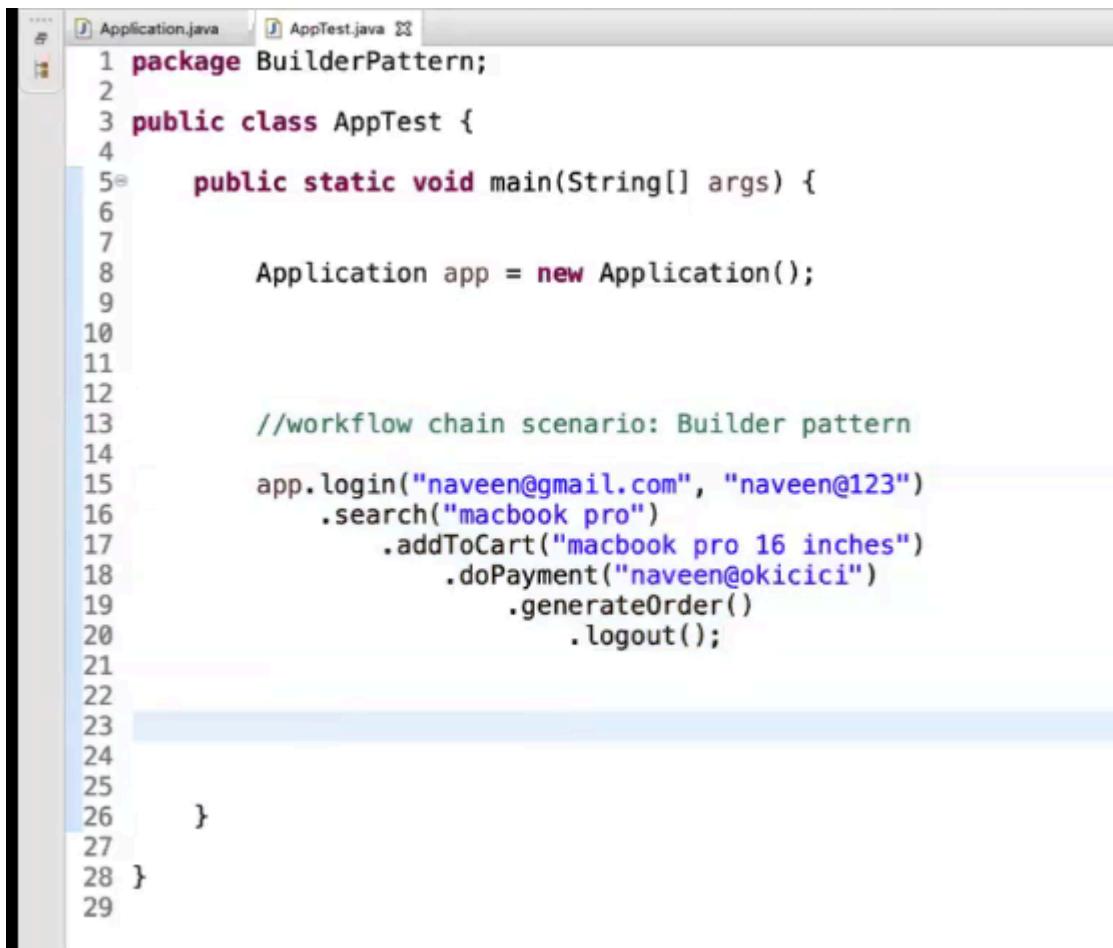


```
    " + pwd);  
  
productName);  
  
ze) {  
productName + " price: " + price);  
  
ne);
```



With one object we are playing around with the entire class.

See builder code in test-



```

1 package BuilderPattern;
2
3 public class AppTest {
4
5     public static void main(String[] args) {
6
7         Application app = new Application();
8
9
10
11
12         //workflow chain scenario: Builder pattern
13
14         app.login("naveen@gmail.com", "naveen@123")
15             .search("macbook pro")
16                 .addToCart("macbook pro 16 inches")
17                     .doPayment("naveen@okicici")
18                         .generateOrder()
19                             .logout();
20
21
22
23
24
25
26     }
27
28 }
29

```

For builder pattern, every method has to return “this”.

Another workflow we created-

```

24
25     app.login("naveen@gmail.com", "naveen@123")
26         .search("tshirt", 1000)
27             .addToCart("nike tshirt")
28                 .logout();|
29

```

Third workflow-

```

28         .logout();
29
30     //3.
31     app.login("naveen@gmail.com", "naveen@123")
32     .search("tshirt", 1000)
33     .addToCart("nike tshirt")
34     .doPayment("7878 8888 9999 7676", 111)
35     .generateOrder()
36     .logout();
37
38
39
40
41
42
43

```

Another workflow-

```

37
38
39     //4.
40     app.login("naveen@gmail.com", "naveen@123")
41     .logout();
42
43
44     //5.
45     app.login("vibha@gmail.com", "vibha@123");
46
47     //6.
48     app.login()
49     .search("keyboard")
50     .addToCart("keyboard apple")
51     .doPayment("vibha@hdfc")
52     .generateOrder()
53     .logout();
54

```

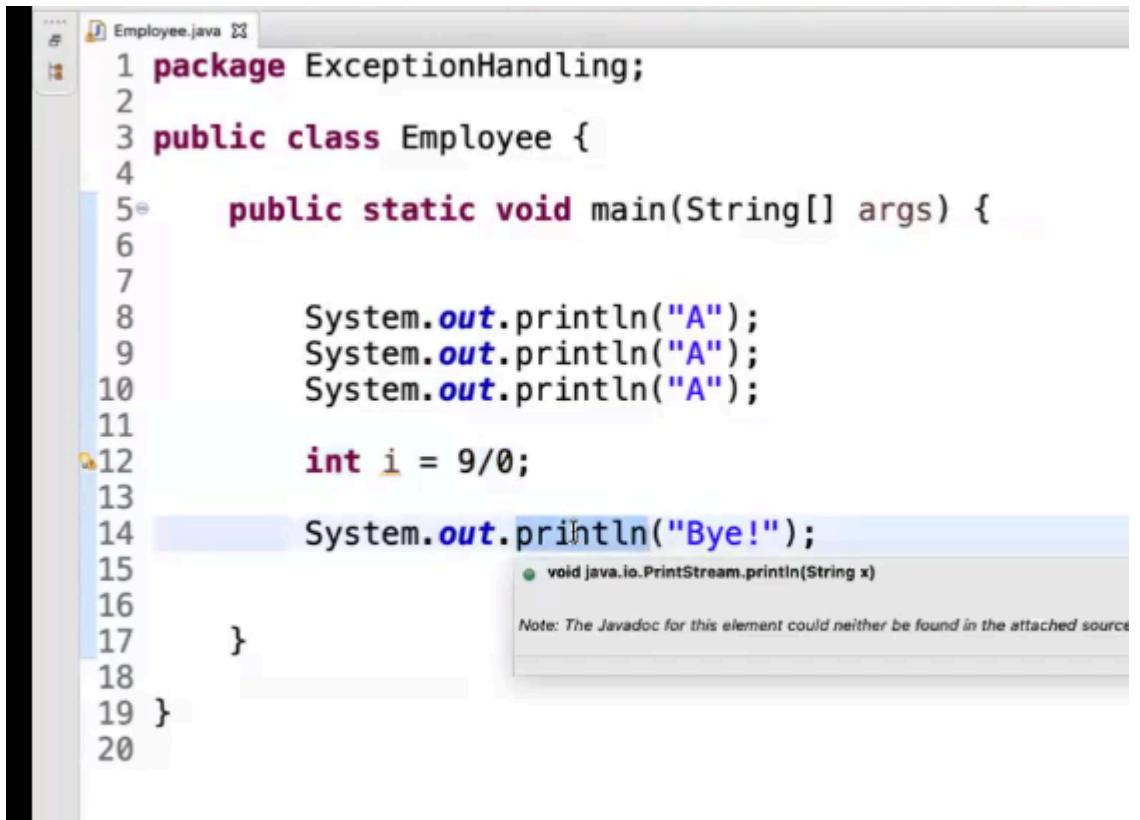
Builder works for one class only at a time. Every class has to have own “return this” statement.
 (26.12)

```

53
54 //this: |
55 //1. to init the class variable within const.. or method
56 //2. can be used with getter/setter: encapsulation
57 //3. same class const.. calling
58 //return this from a function: builder pattern
59
60

```

Exception-

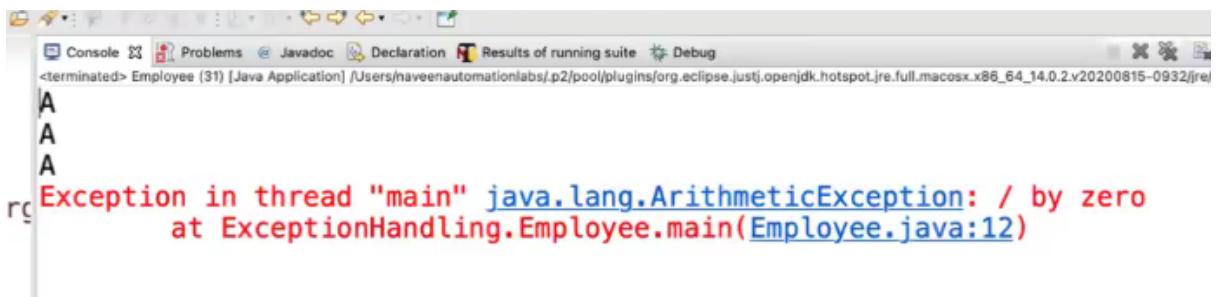


The screenshot shows the Eclipse IDE interface with the Employee.java file open in the editor. The code contains several System.out.println("A") statements and a division by zero error at line 12. A tooltip for the println method is displayed, and the JavaDoc note indicates no attached source.

```

1 package ExceptionHandling;
2
3 public class Employee {
4
5     public static void main(String[] args) {
6
7
8         System.out.println("A");
9         System.out.println("A");
10        System.out.println("A");
11
12        int i = 9/0;
13
14        System.out.println("Bye!");
15
16    }
17
18
19 }
20

```



The screenshot shows the Eclipse IDE interface with the Console tab selected. The output window displays the printed 'A' values followed by an error message indicating an ArithmeticException due to division by zero.

```

[A
A
A
Exception in thread "main" java.lang.ArithmetricException: / by zero
at ExceptionHandling.Employee.main(Employee.java:12)

```

Try cannot come alone-

//Syntax error, insert "Finally" to complete BlockStatements

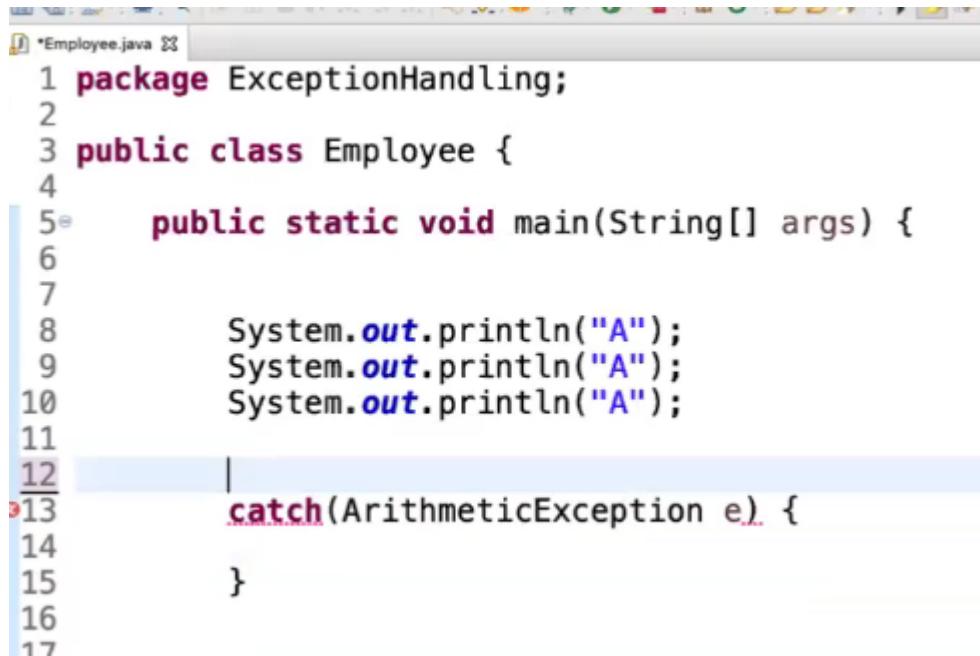
```

1 package ExceptionHandling;
2
3 public class Employee {
4
5     public static void main(String[] args) {
6
7
8         System.out.println("A");
9         System.out.println("A");
10        System.out.println("A");
11
12        try {
13            int i = 9/0;//AE
14        }
15

```

Catch cannot come alone-

//Syntax error on token "catch", invalid RecordHeaderName



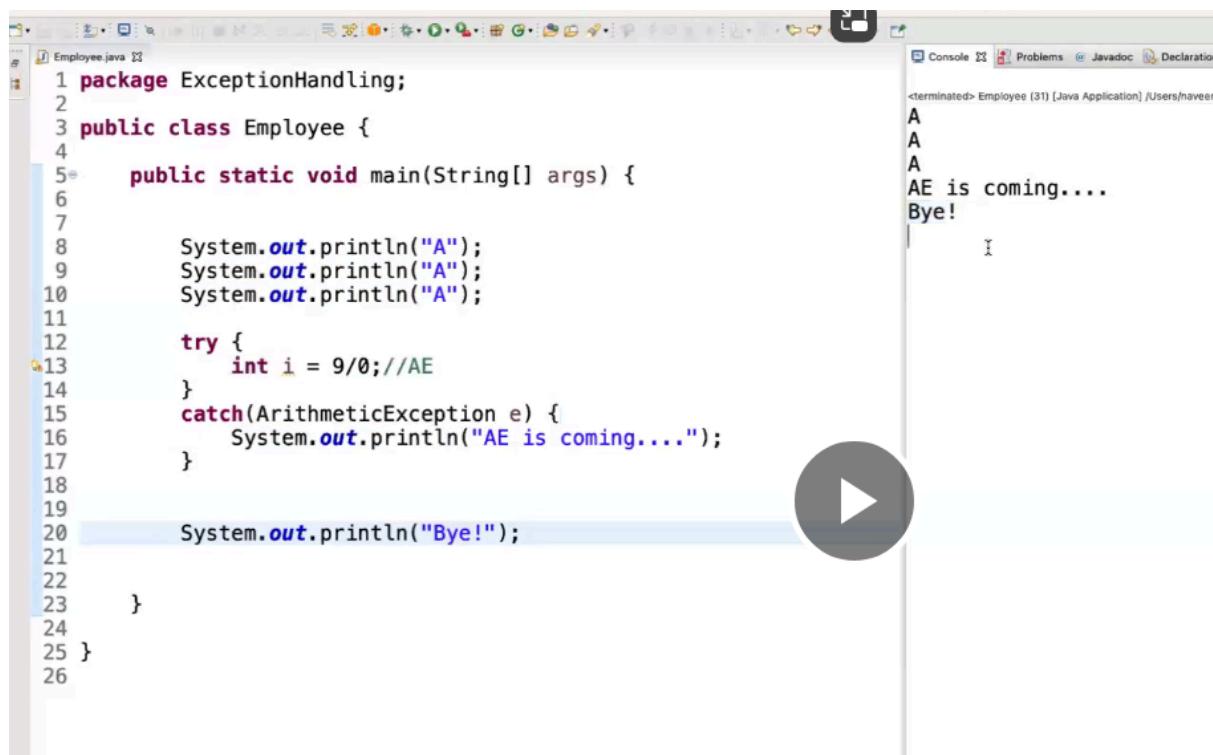
The screenshot shows a Java code editor with the file 'Employee.java' open. The code is identical to the one above, but the 'try' block is followed by a single 'catch' block on the next line:

```

1 package ExceptionHandling;
2
3 public class Employee {
4
5     public static void main(String[] args) {
6
7
8         System.out.println("A");
9         System.out.println("A");
10        System.out.println("A");
11
12        |
13        catch(ArithmetricException e) {
14
15        }
16
17

```

The word 'ArithmetricException' is underlined with a red squiggly line, indicating a spelling error. The cursor is positioned at the start of the 'catch' block.



The screenshot shows an IDE interface with a Java file named Employee.java open. The code contains a main method that prints three 'A's to the console, then attempts to divide by zero in a try block, which catches an ArithmeticException and prints 'AE is coming....'. Finally, it prints 'Bye!'. The console output shows the three 'A's, the exception message, and 'Bye!'.

```

1 package ExceptionHandling;
2
3 public class Employee {
4
5     public static void main(String[] args) {
6
7         System.out.println("A");
8         System.out.println("A");
9         System.out.println("A");
10
11        try {
12            int i = 9/0;//AE
13        }
14        catch(ArithmeticException e) {
15            System.out.println("AE is coming....");
16        }
17
18
19        System.out.println("Bye!");
20
21
22    }
23
24
25 }
26

```

Print stack trace-

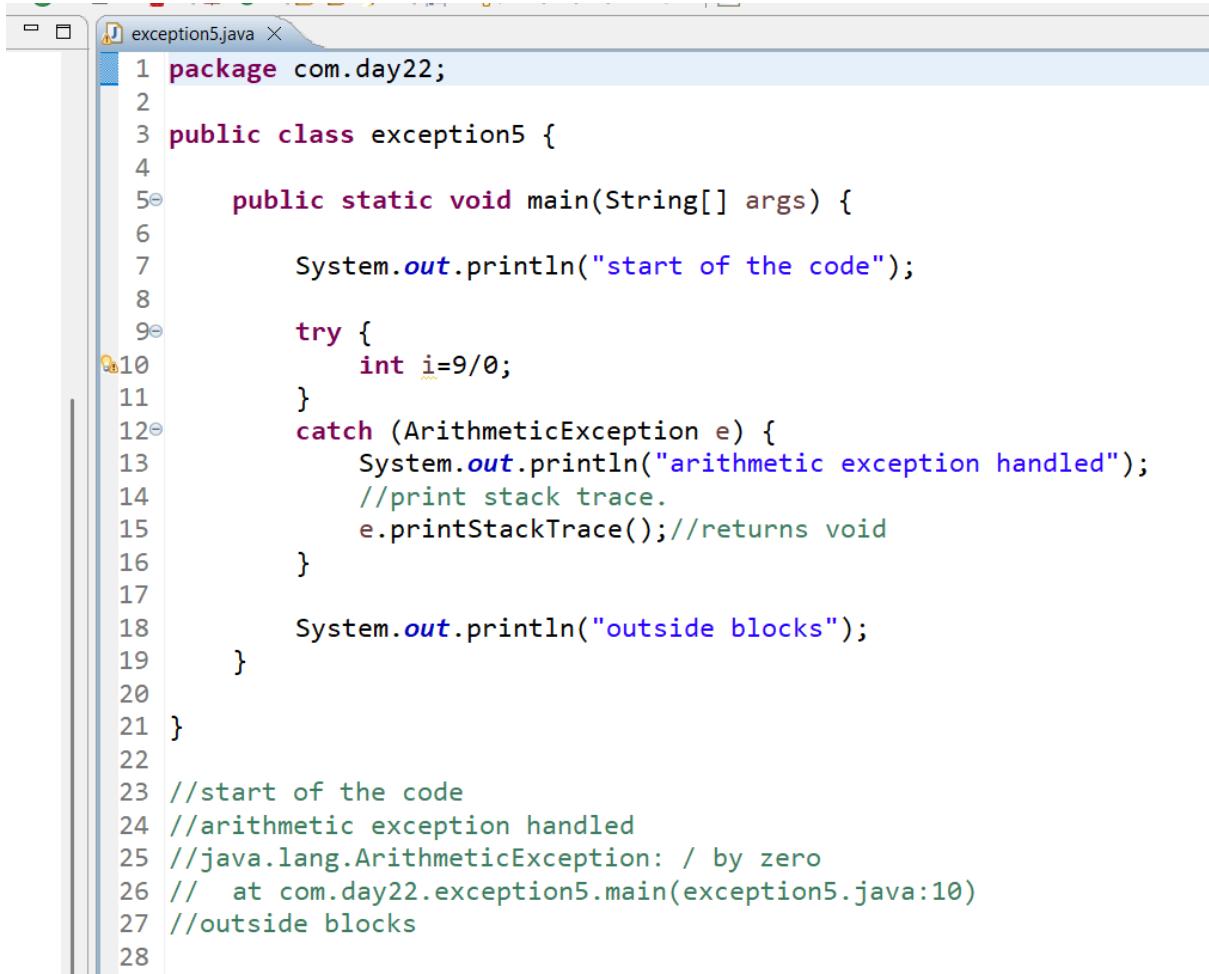
```

14
15      }
16      catch(ArithmeticException e) {
17          System.out.println("AE is coming....");
18          e.printStackTrace();
19

```

java.lang.ArithmetiException: / by zero
 at ExceptionHandling.Employee.main(Employee.java:13)

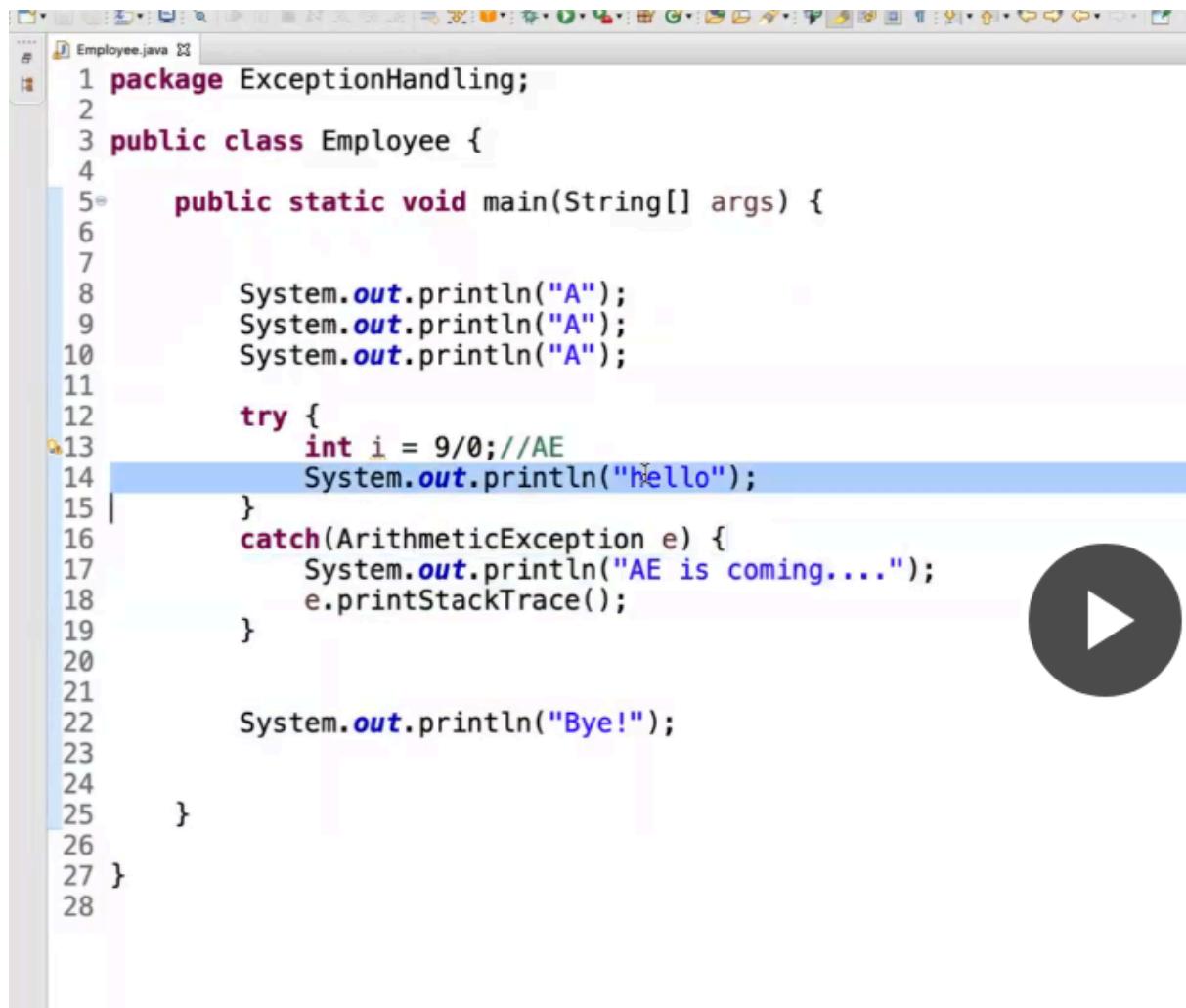
paste exception5-



The screenshot shows a Java code editor with the file "exception5.java" open. The code demonstrates exception handling. It prints "start of the code", attempts to divide by zero, catches the resulting ArithmeticException, prints a message, and then prints "outside blocks". The code is annotated with comments explaining the output.

```
1 package com.day22;
2
3 public class exception5 {
4
5     public static void main(String[] args) {
6
7         System.out.println("start of the code");
8
9         try {
10            int i=9/0;
11        }
12        catch (ArithmaticException e) {
13            System.out.println("arithmetic exception handled");
14            //print stack trace.
15            e.printStackTrace(); //returns void
16        }
17
18        System.out.println("outside blocks");
19    }
20
21 }
22
23 //start of the code
24 //arithmetic exception handled
25 //java.lang.ArithmaticException: / by zero
26 // at com.day22.exception5.main(exception5.java:10)
27 //outside blocks
28
```

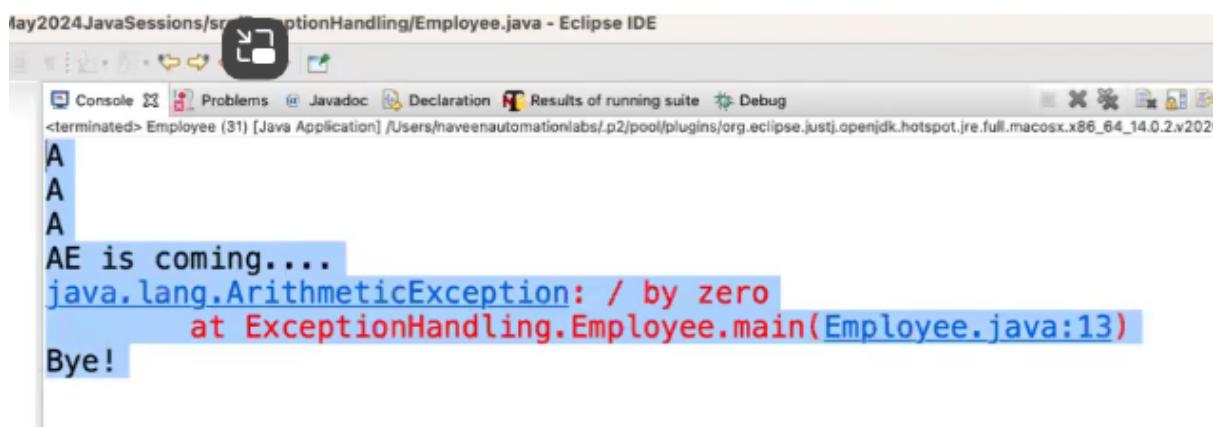
after exception no line runs in try-



```

1 package ExceptionHandling;
2
3 public class Employee {
4
5     public static void main(String[] args) {
6
7         System.out.println("A");
8         System.out.println("A");
9         System.out.println("A");
10
11        try {
12            int i = 9/0;//AE
13            System.out.println("Hello");
14        }
15        catch(ArithmaticException e) {
16            System.out.println("AE is coming....");
17            e.printStackTrace();
18        }
19
20
21        System.out.println("Bye!");
22
23
24    }
25
26
27 }
28

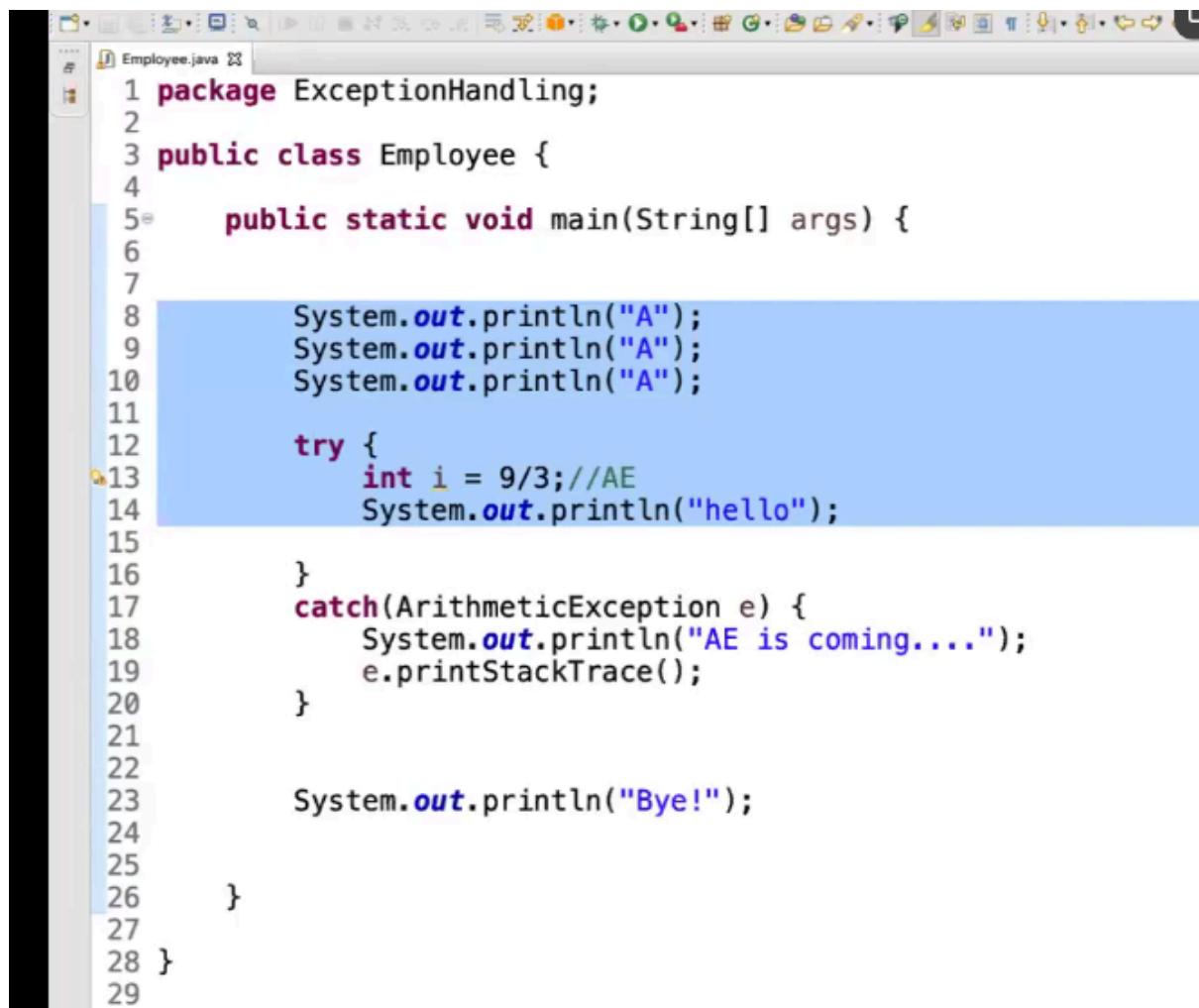
```



May2024JavaSessions/src/ExceptionHandling/Employee.java - Eclipse IDE

A
A
A
AE is coming....
java.lang.ArithmaticException: / by zero
at ExceptionHandling.Employee.main(Employee.java:13)
Bye!

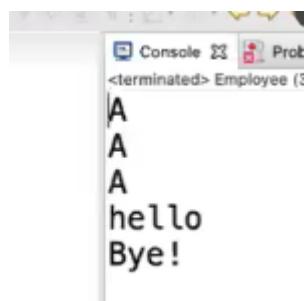
No exception-



```

1 package ExceptionHandling;
2
3 public class Employee {
4
5     public static void main(String[] args) {
6
7         System.out.println("A");
8         System.out.println("A");
9         System.out.println("A");
10
11     try {
12         int i = 9/3;//AE
13         System.out.println("hello");
14     }
15
16     catch(ArithmaticException e) {
17         System.out.println("AE is coming....");
18         e.printStackTrace();
19     }
20
21
22     System.out.println("Bye!");
23
24
25
26 }
27
28 }
29

```



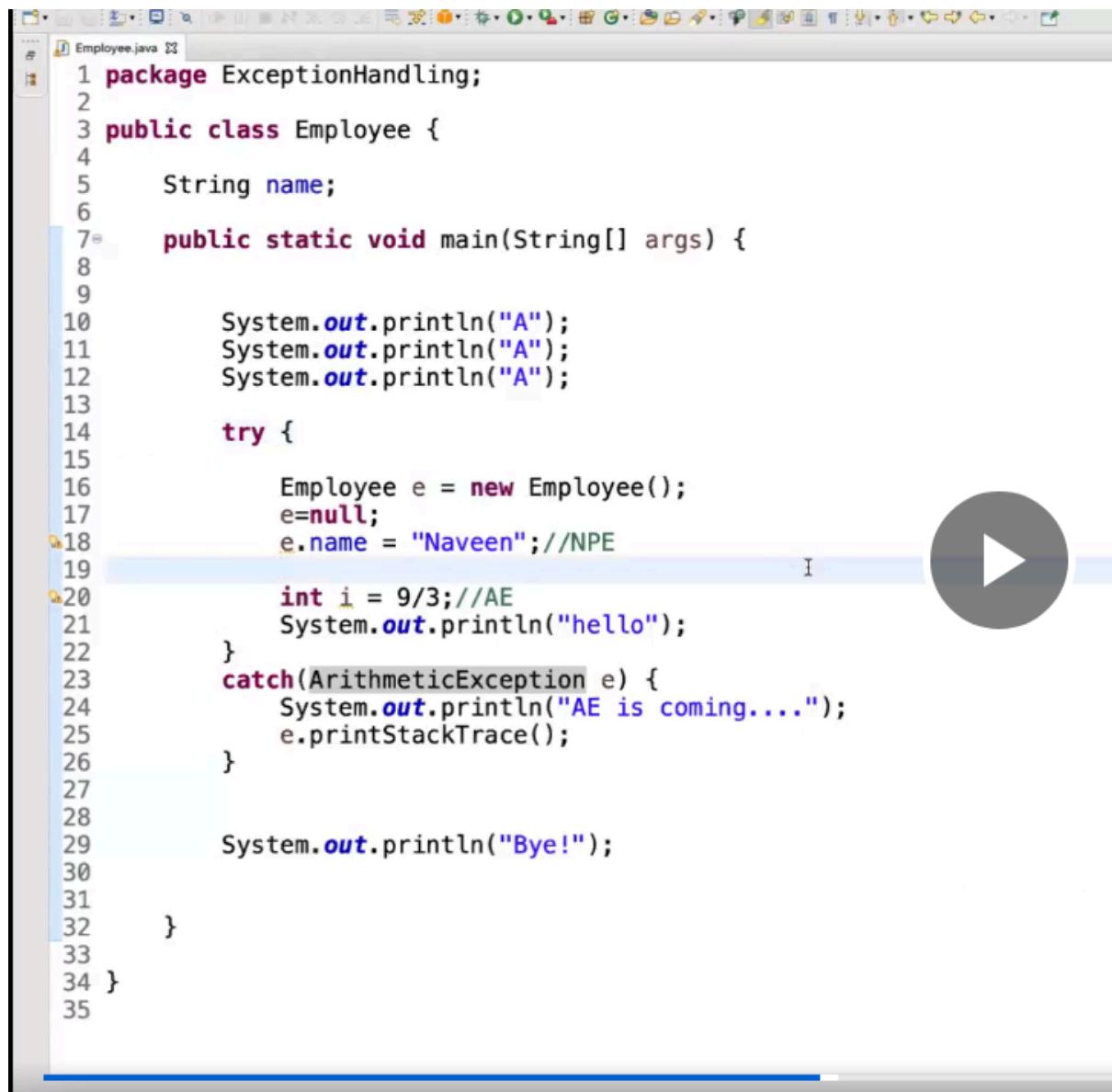
```

Console 23 Problems
<terminated> Employee :1
A
A
A
hello
Bye!

```

Wont catch one(npe) error- 43.51→

Only one exception captured(arithmetic)



```

1 package ExceptionHandling;
2
3 public class Employee {
4
5     String name;
6
7     public static void main(String[] args) {
8
9
10        System.out.println("A");
11        System.out.println("A");
12        System.out.println("A");
13
14        try {
15
16            Employee e = new Employee();
17            e=null;
18            e.name = "Naveen";//NPE
19
20            int i = 9/3;//AE
21            System.out.println("hello");
22        }
23        catch(ArithmaticException e) {
24            System.out.println("AE is coming....");
25            e.printStackTrace();
26        }
27
28
29        System.out.println("Bye!");
30
31    }
32
33}
34
35

```



- May2024JavaSessions/src/ExceptionHandling/Employee.java - Eclipse IDE

Console Problems Javadoc Declaration Results of running suite Debug

<terminated> Employee (31) [Java Application] /Users/naveenautomationlabs/p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_14.0.2.v20200815-0932/jre/bin/java (01-)

A
A
A
Exception in thread "main" java.lang.NullPointerException: Cannot assign file
at ExceptionHandling.Employee.main(Employee.java:18)

Add for npe-

```
21     System.out.println("hello");
22 }
23 catch(ArithmetricException e) {
24     System.out.println("AE is coming....");
25     e.printStackTrace();
26 }
27 catch(NullPointerException e) {
28     System.out.println("NPE is coming....");
29     e.printStackTrace();
30 }
31
32
```

- May2024JavaSessions/src/ExceptionHandling/Employee.java - Eclipse IDE

A
A
A
NPE is coming....
java.lang.NullPointerException: Cannot assign field "name" because "e" is null
at ExceptionHandling.Employee.main(Employee.java:18)
Bye!

paste exception10-

```

exception10.java X
1 package com.day22;
2
3 public class exception10 {
4
5     String name;
6
7     //both exceptions come.
8     //after the first exception rest of the code in try ignored.
9
10    public static void main(String[] args) {
11
12        System.out.println("start of the code");
13
14        try {
15
16            exception10 e1=new exception10();
17            e1=null;
18            e1.name=null;
19            e1.name="karan";
20            System.out.println(e1.name);
21
22            int i=9/0;
23            System.out.println("after exception try block");
24        }
25        catch (ArithmaticException e) {
26            System.out.println("arithmetic exception handled");
27            e.printStackTrace();
28        }
29
30        e.printStackTrace();
31
32        catch (NullPointerException e) {
33            System.out.println("null pointer exception handled");
34            e.printStackTrace();
35        }
36        System.out.println("outside blocks");
37    }
38 }
39
40 //start of the code
41 //null pointer exception handled
42 //java.lang.NullPointerException: Cannot assign field "name" because "e1" is null
43 // at com.day22.exception10.main(exception10.java:17)
44 //outside blocks
45

```

paste exception11-

```

1 package com.day22;
2
3 public class exception11 {
4
5     String name;
6
7     //both exceptions come.
8     //have exceptions in different try blocks.
9
10    public static void main(String[] args) {
11
12        System.out.println("start of the code");
13
14        try {
15
16            exception11 e1=new exception11();
17            e1=null;
18            e1.name=null;
19            e1.name="karan";
20            System.out.println(e1.name);
21        }
22        catch (NullPointerException e) {
23            System.out.println("null pointer exception handled");
24            e.printStackTrace();
25        }
26        try {
27            int i=9/0;
28            System.out.println("after exception try block");
29        }
30        catch (ArithmaticException e) {
31            System.out.println("arithmetic exception handled");
32            e.printStackTrace();
33        }
34
35
36        System.out.println("outside blocks");
37    }
38
39
40 }
41
42 //start of the code
43 //null pointer exception handled
44 //java.lang.NullPointerException: Cannot assign field "name" because "e1" is null
45 // at com.day22.exception11.main(exception11.java:18)
46 //arithmetic exception handled
47 //java.lang.ArithmaticException: / by zero
48 // at com.day22.exception11.main(exception11.java:27)
49 //outside blocks
50

```

Normal code without exception-

The screenshot shows a Java code editor with the file `Employee.java` open. The code is as follows:

```
1 package ExceptionHandling;
2
3 public class Employee {
4
5     String name;
6
7     public static void main(String[] args) {
8
9
10        System.out.println("A");
11        System.out.println("A");
12        System.out.println("A");
13
14        try {
15
16            Employee e = new Employee();
17            //e=null;
18            e.name = "Naveen";//NPE
19
20            int i = 9/3;//AE
21            System.out.println("hello");
22        }
23        catch(ArithmaticException e) {
24            System.out.println("AE is coming....");
25            e.printStackTrace();
26        }
27        catch(NullPointerException e) {
28            System.out.println("NPE is coming....");
29            e.printStackTrace();
30        }
31
32
33        System.out.println("Bye!");
34
35    }
36
37}
```

A tooltip at the bottom of the editor window states: "The value of the local variable `i` is not used".

```

space1 - May2024JavaSessions/src/ExceptionHandling/Employee.java
Console Problems Javadoc Declarations
<terminated> Employee (31) [Java Application] /Users/naveen
A
A
A
hello
Bye!
{

```

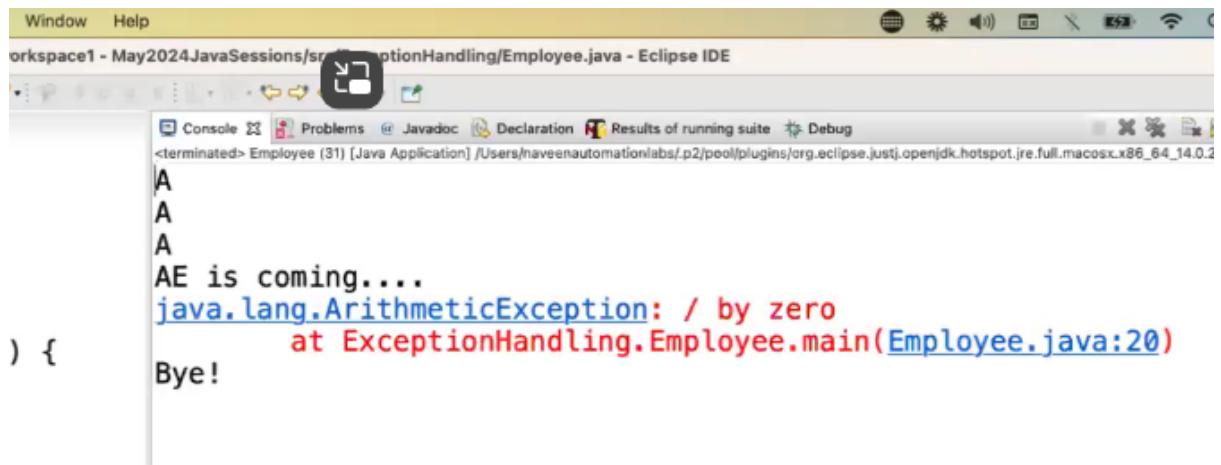
catch arith exception not null -

```

1 package ExceptionHandling;
2
3 public class Employee {
4
5     String name;
6
7     public static void main(String[] args) {
8
9
10    System.out.println("A");
11    System.out.println("A");
12    System.out.println("A");
13
14    try {
15
16        Employee e = new Employee();
17        //e=null;
18        e.name = "Naveen";//NPE
19
20        int i = 9/0;//AE
21        System.out.println("hello");
22    }
23    catch(ArithmeticException e) {
24        System.out.println("AE is coming....");
25        e.printStackTrace();
26    }
27    catch(NullPointerException e) {
28        System.out.println("NPE is coming....");
29        e.printStackTrace();
30    }
31
32
33    System.out.println("Bye!");
34
35
36 }
37

```

The value of the local variable i is not used

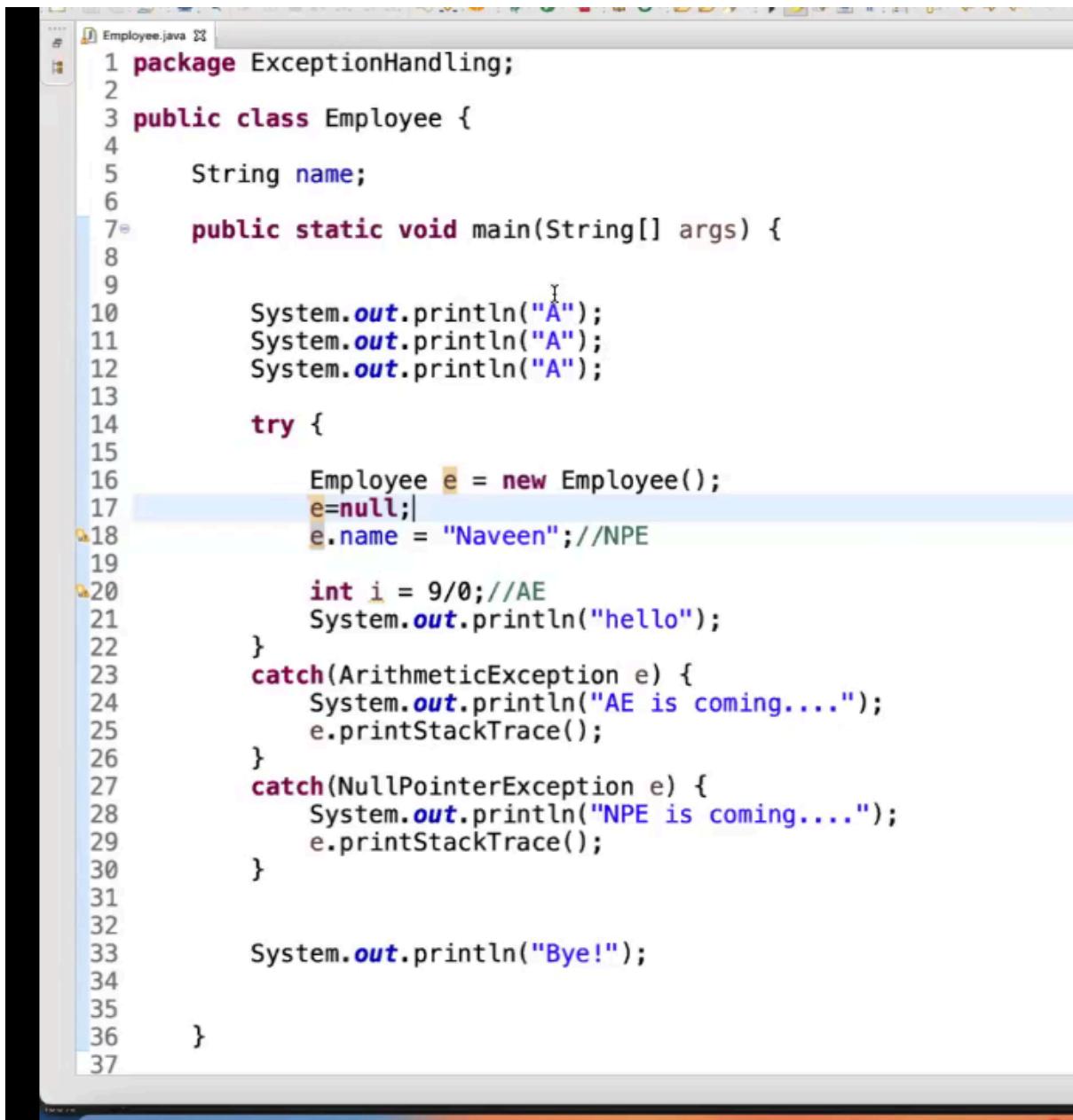


```
A  
A  
A  
AE is coming....  
java.lang.ArithmaticException: / by zero  
at ExceptionHandling.Employee.main(Employee.java:20)  
Bye!
```

At a time only one exception can be thrown because execution happens line by line and the moment first exception comes all other lines not executed below it. All lines from catch and below catch will run.

two exceptions at once-

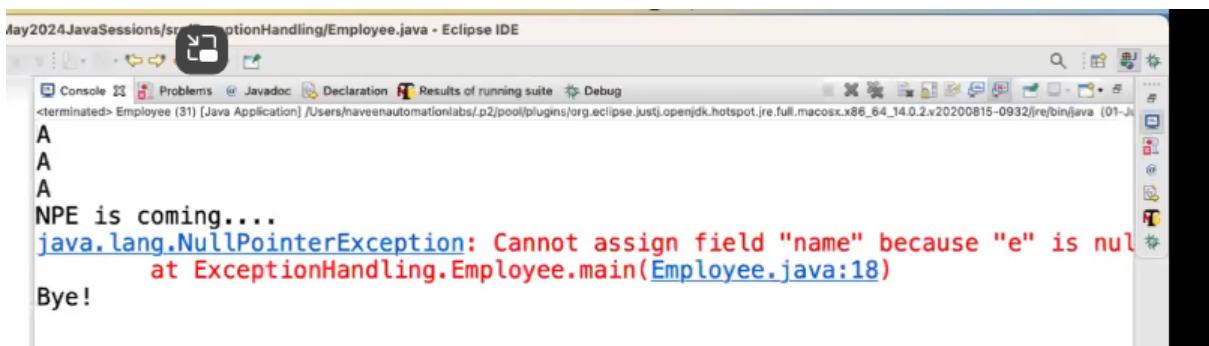
the first one will be caught and program goes ahead.



```

1 package ExceptionHandling;
2
3 public class Employee {
4
5     String name;
6
7     public static void main(String[] args) {
8
9
10        System.out.println("A");
11        System.out.println("A");
12        System.out.println("A");
13
14    try {
15
16        Employee e = new Employee();
17        e=null;
18        e.name = "Naveen";//NPE
19
20        int i = 9/0;//AE
21        System.out.println("hello");
22    }
23    catch(ArithmaticException e) {
24        System.out.println("AE is coming....");
25        e.printStackTrace();
26    }
27    catch(NullPointerException e) {
28        System.out.println("NPE is coming....");
29        e.printStackTrace();
30    }
31
32
33    System.out.println("Bye!");
34
35
36}
37

```



May2024JavaSessions/src/ExceptionHandling/Employee.java - Eclipse IDE

A
A
A
NPE is coming....
java.lang.NullPointerException: Cannot assign field "name" because "e" is null
at ExceptionHandling.Employee.main(Employee.java:18)
Bye!

Parent exception instead of multiple catch blocks-

```

 5     String name;
 6
 7     public static void main(String[] args) {
 8
 9
10         System.out.println("A");
11         System.out.println("A");
12         System.out.println("A");
13
14     try {
15
16         Employee e = new Employee();
17         e=null;
18         e.name = "Naveen";//NPE
19
20         int i = 9/0;//AE
21         System.out.println("hello");
22     }
23     catch(Exception e) {
24         System.out.println("some exception is coming");
25         e.printStackTrace();
26     }
27
28
29 //     catch(ArithmaticException e) {
30 //         System.out.println("AE is coming....");
31 //         e.printStackTrace();
32 //     }
33 //     catch(NullPointerException e) {
34 //         System.out.println("NPE is coming....");
35 //         e.printStackTrace();
36 //     }
37
38
39     System.out.println("Bye!");
40
41

```

Console Problems Javadoc Declaration Results of running suite Debug

<terminated> Employee (31) [Java Application] /Users/naveenautomationlabs/p2pool/plugins/org.eclipse.jdt.openjdk.hotspot.jre.full.macosx.x86_64_14.0.2.v20200815-0932/re/bin/java (01-J...

A
A
A
some exception is coming
java.lang.NullPointerException: Cannot assign field "name" because "e" is null
at ExceptionHandling.Employee.main(Employee.java:18)
Bye!

paste exception14-

```

exception14.java X
1 package com.day22;
2
3 public class exception14 {
4
5     String name;
6
7     //write master class exceptions for any exceptions.
8     //one catch will be enough.
9
10    public static void main(String[] args) {
11
12        System.out.println("start of the code");
13
14        try {
15
16            exception14 e1=new exception14();
17            e1=null;
18            e1.name=null;
19            System.out.println(e1.name);
20        }
21        catch (Exception e) {
22            System.out.println("null pointer exception handled");
23            e.printStackTrace();
24        }
25        try {
26            int i=9/0;
27            System.out.println("after exception try block");
28        }
29        catch (Exception e) {
30            System.out.println("arithmetic exception handled");
31            e.printStackTrace();
32        }
33
34
35        System.out.println("outside blocks");
36    }
37
38 }
39
40
41 //start of the code
42 //null pointer exception handled
43 //java.lang.NullPointerException: Cannot assign field "name" because "e1" is null
44 // at com.day22.exception14.main(exception14.java:18)
45 //arithmetic exception handled
46 //java.lang.ArithmaticException: / by zero
47 // at com.day22.exception14.main(exception14.java:26)
48 //outside blocks
49

```

Master class is “exception” for all exceptions-

```

5 String name;
6
7 public static void main(String[] args) {
8
9     System.out.println("A");
10    System.out.println("A");
11    System.out.println("A");
12
13    try {
14
15        Employee e = new Employee();
16        //e=null;
17        e.name = "Naveen";//NPE
18
19        int i = 9/0;//AE
20        System.out.println("hello");
21    }
22    catch(Exception e) {
23        System.out.println("some exception is coming");
24        e.printStackTrace();
25    }
26
27
28    catch(ArithmeticException e) {
29        System.out.println("AE is coming....");
30        e.printStackTrace();
31    }
32    catch(NullPointerException e) {
33        System.out.println("NPE is coming....");
34        e.printStackTrace();
35    }
36
37
38    System.out.println("Bye!");
39
40
41

```

```

A
A
A
some exception is coming
java.lang.ArithmetricException: / by zero
at ExceptionHandling.Employee.main(Employee.java:20)
Bye!

```

paste exception16-

```

1 package com.day22;
2
3 public class exception16 {
4
5     String name;
6
7     //catch arithmetic using exception.
8
9     public static void main(String[] args) {
10
11         System.out.println("start of the code");
12
13         try {
14
15             exception16 e1=new exception16();
16             e1=null;
17             e1.name="karan";
18             System.out.println(e1.name);
19             int i=9/0;
20             System.out.println("after exception try block");
21         }
22         catch (Exception e) {
23             System.out.println("Catch block with exception");
24             e.printStackTrace();
25         }
26
27         System.out.println("outside blocks");
28     }
29
30 }
31
32 //start of the code
33 //Catch block with exception
34 //java.lang.NullPointerException: Cannot assign field "name" because "e1" is null
35 // at com.day22.exception16.main(exception16.java:17)
36 //outside blocks
37

```

Best practice is multiple catch blocks.

Throwable is super class of exception.

Handling using throwable-

```

4     String name;
5
6     public static void main(String[] args) {
7
8
9         System.out.println("A");
10        System.out.println("A");
11        System.out.println("A");
12
13    try {
14
15        Employee e = new Employee();
16        e=null;
17        e.name = "Naveen";//NPE
18
19        int i = 9/0;//AE
20        System.out.println("hello");
21    }
22    catch(Throwable e) {
23        System.out.println("some exception is coming");
24        e.printStackTrace();
25    }
26
27    //    catch(ArithmeticException e) {
28    //        System.out.println("AE is coming....");
29    //        e.printStackTrace();
30    //    }
31    //    catch(NullPointerException e) {
32    //        System.out.println("NPE is coming....");
33    //        e.printStackTrace();
34    //    }
35
36
37
38    System.out.println("Bye!");
39
40

```

```

A
A
A
some exception is coming
java.lang.NullPointerException: Cannot assign field "name" because "e" is null
at ExceptionHandling.Employee.main(Employee.java:18)
Bye!

```

Object is super class of all class including throwable.

It will throw error when we use object for catching exceptions-

//No exception of type Object can be thrown;

//an exception type must be a subclass of Throwable

```

13
14     try {
15
16         Employee e = new Employee();
17         e=null;
18         e.name = "Naveen";//NPE
19
20         int i = 9/0;//AE
21         System.out.println("hello");
22     }
23     catch(Object e) {
24         System.out.println("some exception is coming");
25         e.printStackTrace();
26     }
27
28 //    catch(ArithmaticException a) {

```



Only throwable or lower than that is allowed.

Cant add the exception above and child exceptions below-

Top one already handling it.

```

//Unreachable catch block for ArithmaticException.
//It is already handled by the catch block for Exception

```

```

13
14     try {
15
16         Employee e = new Employee();
17         e=null;
18         e.name = "Naveen";//NPE
19
20         int i = 9/0;//AE
21         System.out.println("hello");
22     }
23     catch(Exception e) {
24         System.out.println("some exception is coming");
25         e.printStackTrace();
26     }
27
28     catch(ArithmeticException e) {
29         System.out.println("AE is coming....");
30         e.printStackTrace();
31     }
32     catch(NullPointerException e) {
33         System.out.println("NPE is coming....");
34         e.printStackTrace();
35     }
36

```



This one is ok-

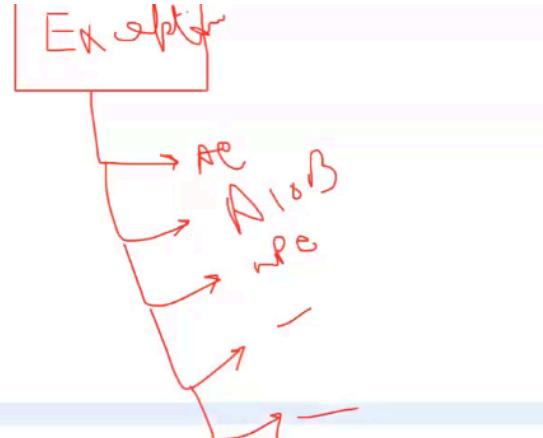
Line by line code runs.

Hence lower one can be at top.

```

13
14     try {
15
16         Employee e = new Employee();
17         e=null;
18         e.name = "Naveen";//NPE
19
20         int i = 9/0;//AE
21         System.out.println("hello");
22     }
23
24     catch(ArithmeticException e) {
25         System.out.println("AE is coming....");
26         e.printStackTrace();
27     }
28     catch(NullPointerException e) {
29         System.out.println("NPE is coming....");
30         e.printStackTrace();
31     }
32
33     catch(Exception e) {
34         System.out.println("some exception is coming");
35         e.printStackTrace();
36     }
37

```



This is also ok-

```

22
23
24
25
26     catch(ArithmaticException e) {
27         System.out.println("AE is coming....");
28         e.printStackTrace();
29     }
30     catch(NullPointerException e) {
31         System.out.println("NPE is coming....");
32         e.printStackTrace();
33     }
34
35     catch(Exception e) {
36         System.out.println("some exception is coming");
37         e.printStackTrace();
38     }
39
40     catch(Throwable e) {
41         System.out.println("some exception is coming");
42         e.printStackTrace();
43     }
44

```

paste exception22-

```

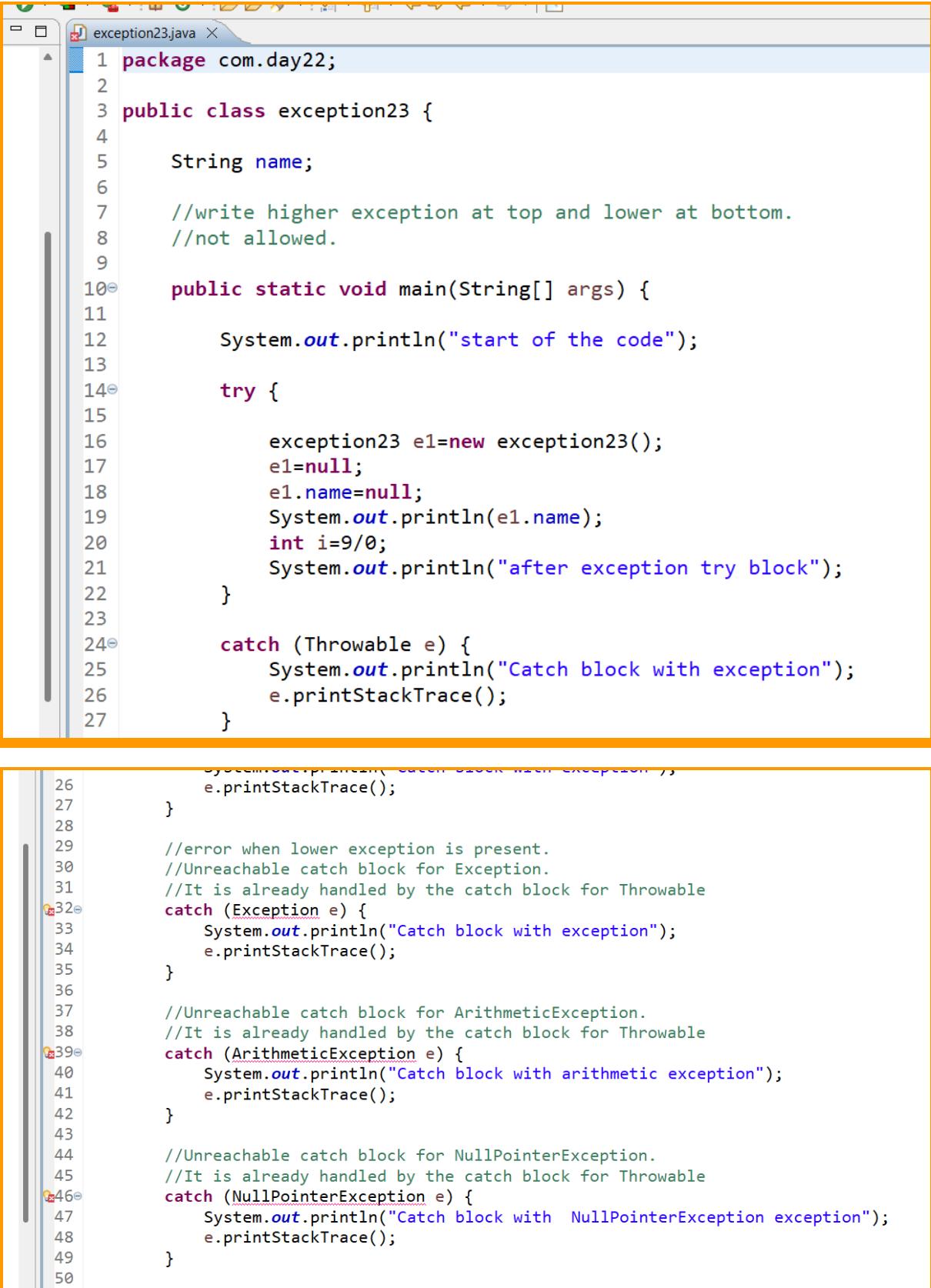
1 package com.day22;
2
3 public class exception22 {
4
5     String name;
6
7     //write higher exception at top and lower at bottom.
8     //not allowed.
9
10    public static void main(String[] args) {
11
12        System.out.println("start of the code");
13
14        try {
15
16            exception22 e1=new exception22();
17            e1=null;
18            e1.name=null;
19            System.out.println(e1.name);
20            int i=9/0;
21            System.out.println("after exception try block");
22        }
23
24        catch (Throwable e) {
25            System.out.println("Catch block with exception");
26            e.printStackTrace();
27        }
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44

```

```
26         e.printStackTrace();
27     }
28
29     //Unreachable catch block for ArithmeticException.
30     //It is already handled by the catch block for Throwable
31     catch (ArithmaticException e) {
32         System.out.println("Catch block with arithmetic exception");
33         e.printStackTrace();
34     }
35
36     //Unreachable catch block for NullPointerException.
37     //It is already handled by the catch block for Throwable
38     catch (NullPointerException e) {
39         System.out.println("Catch block with NullPointerException exception");
40         e.printStackTrace();
41     }
42
43
44
45     System.out.println("outside blocks");
46 }
```

```
45     System.out.println("outside blocks");
46 }
47
48 }
49
50 //start of the code
51 //Catch block with NullPointerException exception
52 //java.lang.NullPointerException: Cannot assign field "name" because "e1" is null
53 // at com.day22.exception21.main(exception21.java:18)
54 //outside blocks
55
```

paste exception23-



The screenshot shows a Java code editor with an orange border. The title bar says "exception23.java". The code is as follows:

```

1 package com.day22;
2
3 public class exception23 {
4
5     String name;
6
7     //write higher exception at top and lower at bottom.
8     //not allowed.
9
10    public static void main(String[] args) {
11
12        System.out.println("start of the code");
13
14        try {
15
16            exception23 e1=new exception23();
17            e1=null;
18            e1.name=null;
19            System.out.println(e1.name);
20            int i=9/0;
21            System.out.println("after exception try block");
22        }
23
24        catch (Throwable e) {
25            System.out.println("Catch block with exception");
26            e.printStackTrace();
27        }
28
29        //error when lower exception is present.
30        //Unreachable catch block for Exception.
31        //It is already handled by the catch block for Throwable
32        catch (Exception e) {
33            System.out.println("Catch block with exception");
34            e.printStackTrace();
35        }
36
37        //Unreachable catch block for ArithmeticException.
38        //It is already handled by the catch block for Throwable
39        catch (ArithmaticException e) {
40            System.out.println("Catch block with arithmetic exception");
41            e.printStackTrace();
42        }
43
44        //Unreachable catch block for NullPointerException.
45        //It is already handled by the catch block for Throwable
46        catch (NullPointerException e) {
47            System.out.println("Catch block with NullPointerException exception");
48            e.printStackTrace();
49        }
50

```

The code demonstrates various catch blocks for different exception types. Lines 32, 39, and 46 are highlighted with orange icons, indicating they are unreachable code.

```

49 }
50
51
52
53     System.out.println("outside blocks");
54 }
55
56 }
57
58 //start of the code
59 //Catch block with NullPointerException exception
60 //java.lang.NullPointerException: Cannot assign field "name" because "e1" is null
61 // at com.day22.exception21.main(exception21.java:18)
62 //outside blocks
63
64

```

paste exception24-

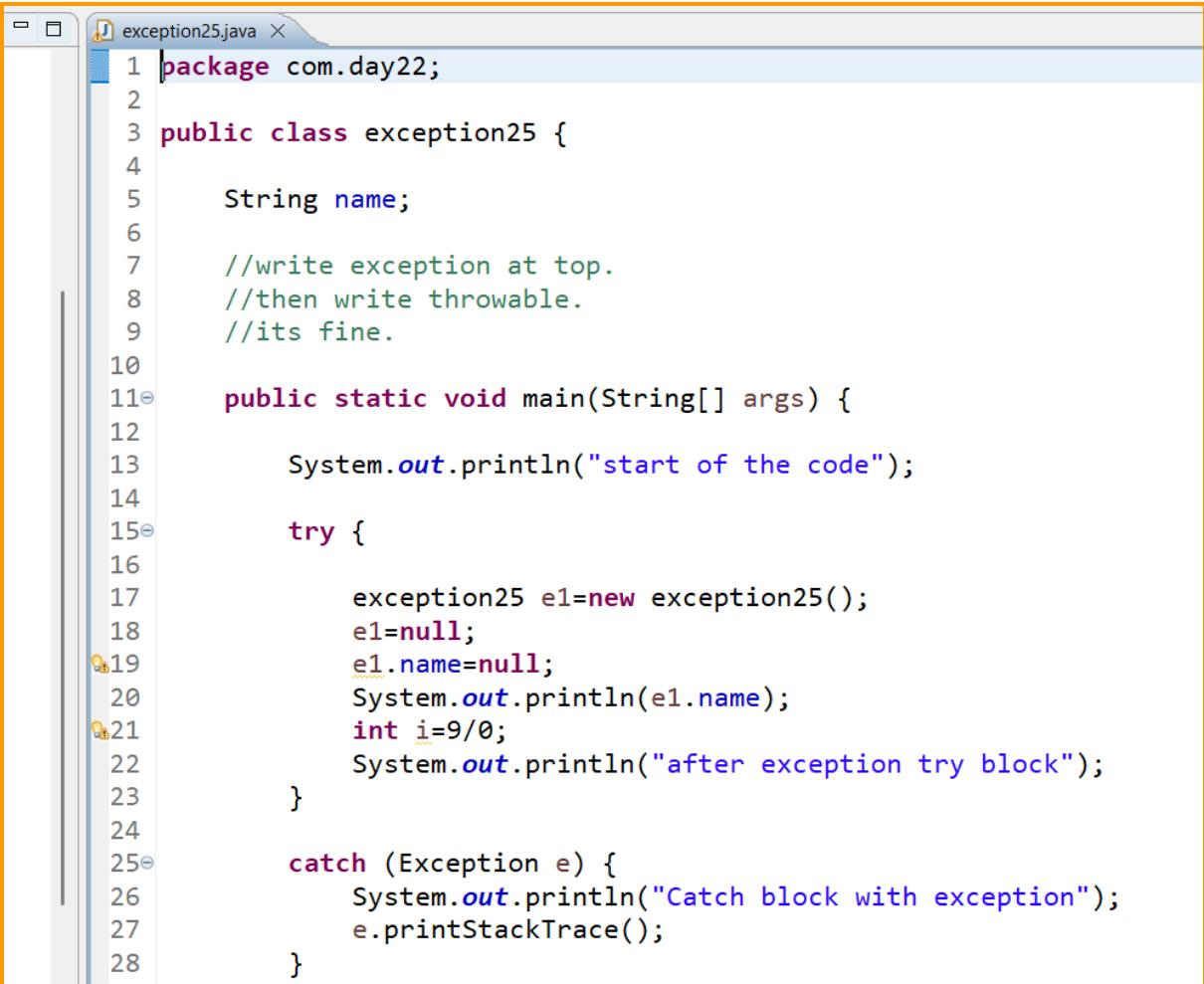
```

1 package com.day22;
2
3 public class exception24 {
4
5     String name;
6
7     //write exception at top.
8     //then write throwable.
9     //its fine.
10
11    public static void main(String[] args) {
12
13        System.out.println("start of the code");
14
15        try {
16
17            exception24 e1=new exception24();
18            e1=null;
19            e1.name=null;
20            System.out.println(e1.name);
21            int i=9/0;
22            System.out.println("after exception try block");
23        }
24
25        catch (Exception e) {
26            System.out.println("Catch block with exception");
27            e.printStackTrace();
28        }

```

```
27         e.printStackTrace();
28     }
29
30     catch (Throwable e) {
31         System.out.println("Catch block with exception");
32         e.printStackTrace();
33     }
34
35     //Unreachable catch block for ArithmeticException.
36     //It is already handled by the catch block for Throwable
37     catch (ArithmetcException e) {
38         System.out.println("Catch block with arithmetic exception");
39         e.printStackTrace();
40     }
41
42     //Unreachable catch block for NullPointerException.
43     //It is already handled by the catch block for Throwable
44     catch (NullPointerException e) {
45         System.out.println("Catch block with NullPointerException exception");
46         e.printStackTrace();
47     }
48
49
50
51     System.out.println("outside blocks");
52 }
53
54 }
55
56 //start of the code
57 //Catch block with NullPointerException exception
58 //java.lang.NullPointerException: Cannot assign field "name" because "e1" is null
59 //  at com.day22.exception21.main(exception21.java:18)
60 //outside blocks
61
```

paste exception25-



```

1 package com.day22;
2
3 public class exception25 {
4
5     String name;
6
7     //write exception at top.
8     //then write throwable.
9     //its fine.
10
11    public static void main(String[] args) {
12
13        System.out.println("start of the code");
14
15        try {
16
17            exception25 e1=new exception25();
18            e1=null;
19            e1.name=null;
20            System.out.println(e1.name);
21            int i=9/0;
22            System.out.println("after exception try block");
23        }
24
25        catch (Exception e) {
26            System.out.println("Catch block with exception");
27            e.printStackTrace();
28        }
29
30        catch (Throwable e) {
31            System.out.println("Catch block with exception");
32            e.printStackTrace();
33        }
34
35        System.out.println("outside blocks");
36    }
37
38 }
39
40 //start of the code
41 //Catch block with exception
42 //java.lang.NullPointerException: Cannot assign field "name" because "e1" is null
43 //  at com.day22.exception25.main(exception25.java:19)
44 //outside blocks

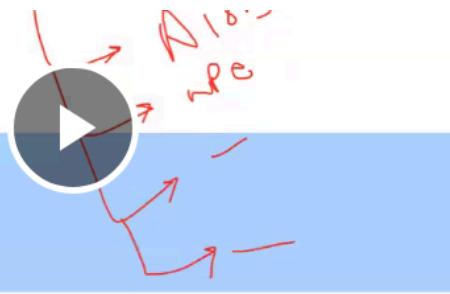
```

This is not allowed-

```

23
24     catch(Throwable e) {
25         System.out.println("some exception is coming");
26         e.printStackTrace();
27     }
28
29     catch(ArithmetricException e) {
30         System.out.println("AE is coming....");
31         e.printStackTrace();
32     }
33     catch(NullPointerException e) {
34         System.out.println("NPE is coming....");
35         e.printStackTrace();
36     }
37
38     catch(Exception e) {
39         System.out.println("some exception is coming");
40         e.printStackTrace();
41     }
42
43

```



Error-

Child of throwable.

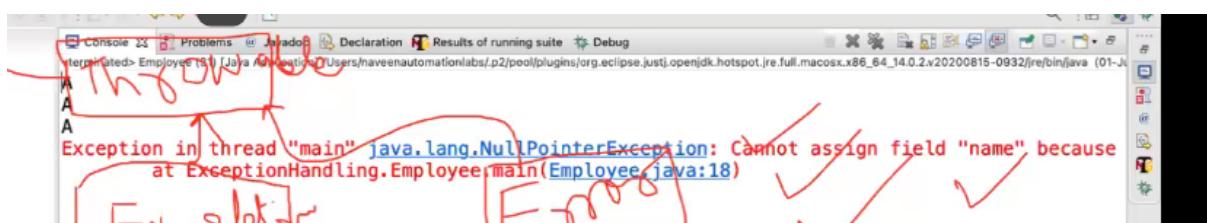
Try handling exception with error-

```

12     System.out.println("A");
13
14     try {
15
16         Employee e = new Employee();
17         e=null;
18         e.name = "Naveen";//NPE
19
20         int i = 9/0;//AE
21         System.out.println("hello");
22     }
23
24     catch(Error e) {
25         System.out.println("some error is coming...");
26         e.printStackTrace();
27     }
28

```

Cannot be handled.



Errors can be handled with try catch but difficult to catch them.

Handled stack exception-



```
Employee.java Student.java
1 package ExceptionHandling;
2
3 public class Student {
4
5     public void m1() {
6         System.out.println("m1");
7         m2();
8     }
9
10    public void m2() {
11        System.out.println("m2");
12        m1();
13    }
14
15    public static void main(String[] args) {
16
17        Student st = new Student();
18
19        try {
20            st.m1();
21        } catch (Error e) {
22            System.out.println("some error is coming...");
23        }
24
25    }
26
27 }
```

The screenshot shows a Java application running in an IDE. The console tab displays the following output:

```
<terminated> Student (11) [Java Application] /Users/haveenautomationlabs/p2/p
m1
m2
some error is coming...
```

The word "error" is partially visible in red, indicating a syntax error or exception. A large play button icon is overlaid on the right side of the console window.

paste exception27-

```

exception27.java X
1 package com.day22;
2
3 public class exception27 {
4
5     //unreachable block in method m2.
6
7     String name;
8
9     public void m1() {
10         System.out.println("m1");
11         int i1=m2(name);
12         System.out.println(i1);
13     }
14
15     public int m2(String name1) {
16         System.out.println("m2 method " + name1);
17         return 10;
18         m1(); //Unreachable code
19     }
20
21     public static void main(String[] args) {
22
23         System.out.println("start of the code");
24     }
}

```



```

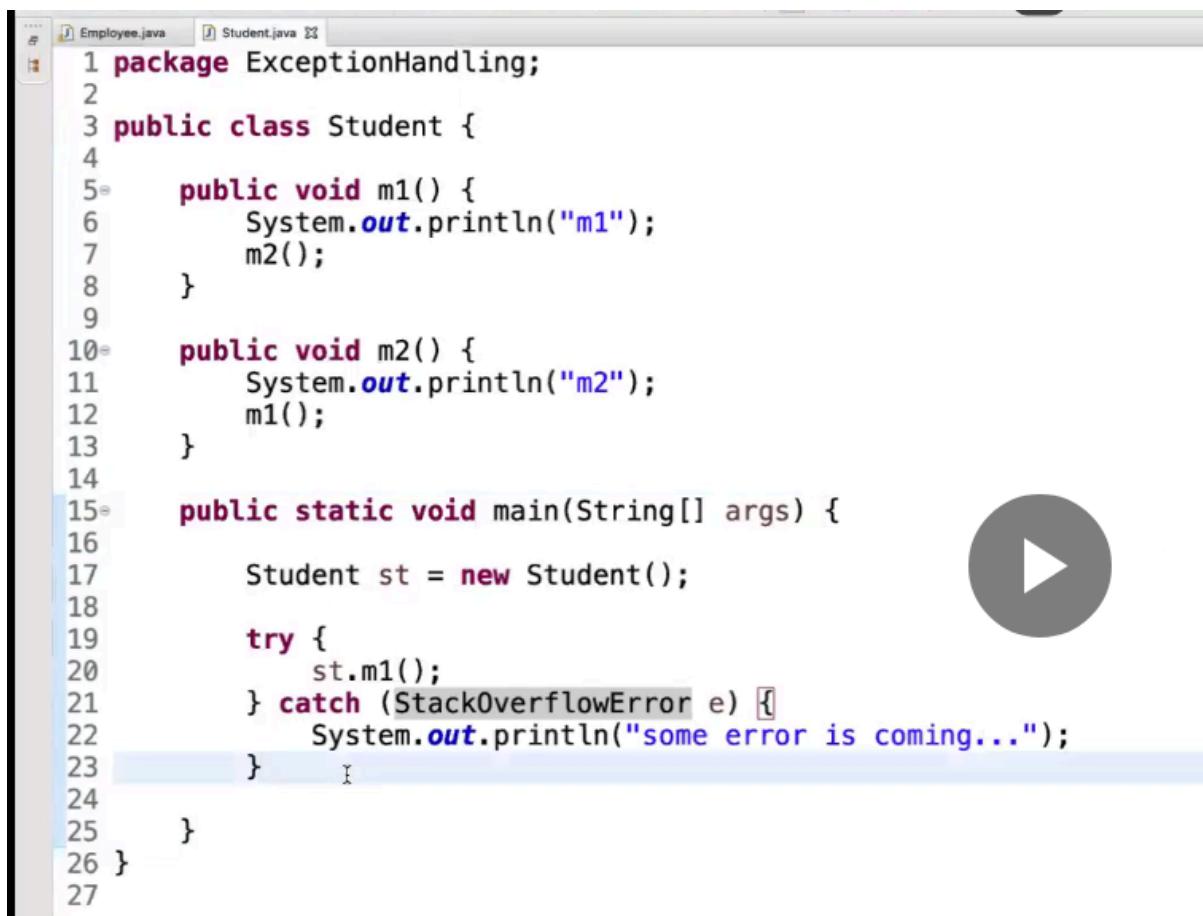
21
22
23         System.out.println("start of the code");
24
25     try {
26
27         exception27 e1=new exception27();
28         e1=null;
29         e1.name=null;
30         System.out.println(e1.name);
31         int i=9/0;
32         System.out.println("after exception try block");
33     }
34
35     catch (Error e) {
36         System.out.println("Catch block with exception");
37         e.printStackTrace();
38     }
39
40         System.out.println("outside blocks");
41     }
42
43 }
44

```

//tried catching error but not easy to debug.

Write exact error name-

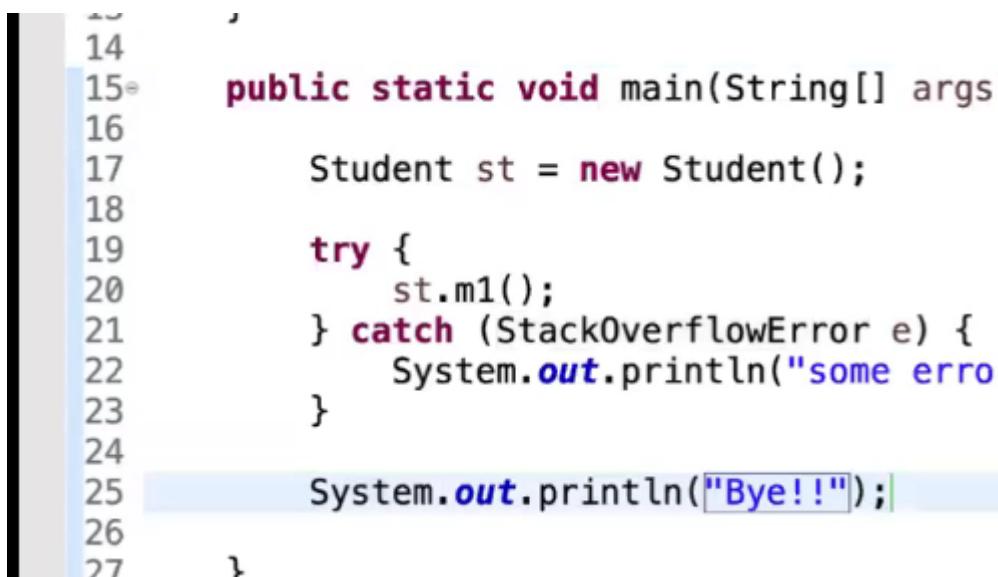




```

1 package ExceptionHandling;
2
3 public class Student {
4
5     public void m1() {
6         System.out.println("m1");
7         m2();
8     }
9
10    public void m2() {
11        System.out.println("m2");
12        m1();
13    }
14
15    public static void main(String[] args) {
16
17        Student st = new Student();
18
19        try {
20            st.m1();
21        } catch (StackOverflowError e) {
22            System.out.println("some error is coming...");
23        }
24
25    }
26
27 }
```

Print bye-



```

14
15    public static void main(String[] args
16
17        Student st = new Student();
18
19        try {
20            st.m1();
21        } catch (StackOverflowError e) {
22            System.out.println("some erro
23        }
24
25        System.out.println("Bye!!");
26
27    }
```



```
m2
m1
m2
some error is coming...
Bye!!
```

Stack will have multiple execution blocks-

One block full then stack error and then second block full will have another stack error and so on.



Can handle exceptions and errors anywhere
where you like-

The screenshot shows a Java IDE interface with the Student.java file open. The code implements exception handling by catching a StackOverflowError. The IDE's status bar indicates the current file is Student.java.

```
1 package ExceptionHandling;
2
3 public class Student {
4
5     public void m1() {
6         System.out.println("m1");
7         m2();
8     }
9
10    public void m2() {
11        System.out.println("m2");
12
13        try {
14            m1();
15        } catch (StackOverflowError e) {
16            System.out.println("some error is coming...");
17        }
18    }
19
20
21    public static void main(String[] args) {
22
23        Student st = new Student();
24
25        st.m1();
26
27        System.out.println("Bye!!");
28    }
29
30 }
31
32
33
34
```



```
<terminated> Student (11) [Java Application] /Users/haveenauto
m2
m1
r2
m1
m2
m1
m2
m1
m2
m1
m2
m1
m2
m1
some error is coming...
Bye!!
```

It keeps throwing error until stack is overflowing.

paste exception30-

```
exception30.java X
1 package com.day22;
2
3 public class exception30 {
4
5     /**
6      * Can handle exceptions and errors anywhere where you like
7      * in the code.
8      * it keeps going in circles and finally nothing after catch block runs
9      * in this case.
10     */
11
12     public void m1() {
13         System.out.println("m1");
14         m2();
15     }
16
17     public void m2() {
18
19         System.out.println("m2");
20         try {
21             m1();
22         }
23         catch (StackOverflowError e) {
24             System.out.println("stack overflow inside method");
25             e.printStackTrace();
26         }
27         //it throws this error for me.
28         //just written for reference.
29     //     catch (NoClassDefFoundError e) {
30     //         System.out.println("no class def found overflow inside method");
31     //         e.printStackTrace();
32     //     }
33
34         System.out.println("m2 method");
35     }
36 }
```

```
--  
34     System.out.println("m2 method");  
35 }  
36  
37 public static void main(String[] args) {  
38  
39     System.out.println("start of the code");  
40  
41     exception30 e1=new exception30();  
42  
43     e1.m1();  
44  
45     System.out.println("outside blocks");  
46 }  
47  
48 }  
49
```

```
48 }
49
50
51 //start of the code
52 //m1
53 //m2
54 //m1
55 //m2
56 //m1
57 //m2
58 //m1
59 //m2
60 //m1
61 //m2
62 //m1
63 //m2
64 //m1
65 //m2
66 //m1
67 //m2
68 //stack overflow inside method
69 //stack overflow inside method
70 //java.lang.StackOverflowError
```

```
69 //stack overflow inside method
70 //java.lang.StackOverflowError
71 //java.lang.StackOverflowError
72 //stack overflow inside method
73 //java.lang.StackOverflowError
74 //stack overflow inside method
75 //java.lang.StackOverflowError
76 //stack overflow inside method
77 //java.lang.StackOverflowError
78 //stack overflow inside method
79 //java.lang.StackOverflowError
80 //stack overflow inside method
81 //java.lang.StackOverflowError
82 //stack overflow inside method
83 //java.lang.StackOverflowError
84 //stack overflow inside method
85 //java.lang.StackOverflowError
86 //stack overflow inside method
87 //java.lang.StackOverflowError
88 //stack overflow inside method
89 //java.lang.StackOverflowError
```

```
88 //stack overflow inside method
89 //java.lang.StackOverflowError
90 //Exception in thread "main" java.lang.NoClassDefFoundError:
91 //Could not initialize class java.lang.StackTraceElement$HashedModules
92 //
93 //Exception: java.lang.NoClassDefFoundError thrown from
94 //the UncaughtExceptionHandler in thread "main"
```