Exception Handling

Handling invalid user input

Produced by: Mairead Meagher
Dr. Siobhán Drohan

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/

Shop V5.0 (or any version)

- When testing it, did you try to enter a **String** instead of an **int**? e.g. for the Product code?

- What happened?

```
Shop Menu
---------
  1) Add a Product
  2) List the Products
  3) Update a Product
  4) Delete a Product
---------
  5) List the cheapest product
  6) List the products in our current product line
  7) Display average product unit cost
  8) List products that are more expensive than a given price
---------
  9) Save Products to product.xml
  10) Load Products from product.xml
---------
  0) Exit
==>> 1
Please enter the product description: Prod01
Please enter the product code: Prod01
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Unknown Source)
        at java.util.Scanner.next(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at java.util.Scanner.nextInt(Unknown Source)
        at Driver.addProduct(Driver.java:126)
        at Driver.runMenu(Driver.java:69)
        at Driver.<init>(Driver.java:18)
        at Driver.main(Driver.java:26)
```

Shop V5.0 (or any version)

- The following code caused a runtime error...

```
int code = input.nextInt();
```

- This is called a **runtime exception**.
- How do we fix this? How do we stop the program from crashing?

What are Exceptions?

- An Exception is an object that signals that some unusual condition has occurred while the program is executing.

- Exceptions are intended to be *detected* and *handled*, so that the program can continue in a sensible way if at all possible.

- Java has many predefined Exception objects, and we can also create our own.

When an exception occurs…

...the normal flow of execution is disrupted and transferred to code, which can handle the exception condition.

The exception mechanism is a lot cleaner than having to check an error value after every method call that could potentially fail.

RuntimeException

- is a subclass of the Exception class
- encompasses all exceptions which can ordinarily happen at run-time.
- these exceptions can be thrown by any java statement or a method call.
- can be avoided through good programming practices!

RuntimeException	Example Causes
ArithmeticException	Can be caused by dividing by zero.
ArrayIndexOutOfBoundsException	Referencing an array index number of 7 when only 5 exist in the array.
NullPointerException	trying to access an object that has no memory allocated yet.

Catching Exceptions

- Catching an exception means declaring that you can handle exceptions of a particular class from a particular block of code.

- You specify the block of code and then provide handlers for various classes of exception.

- If an exception occurs then execution transfers to the corresponding piece of handler code.

try and catch

To catch exceptions, you surround a block of code with a "try, catch" statement.

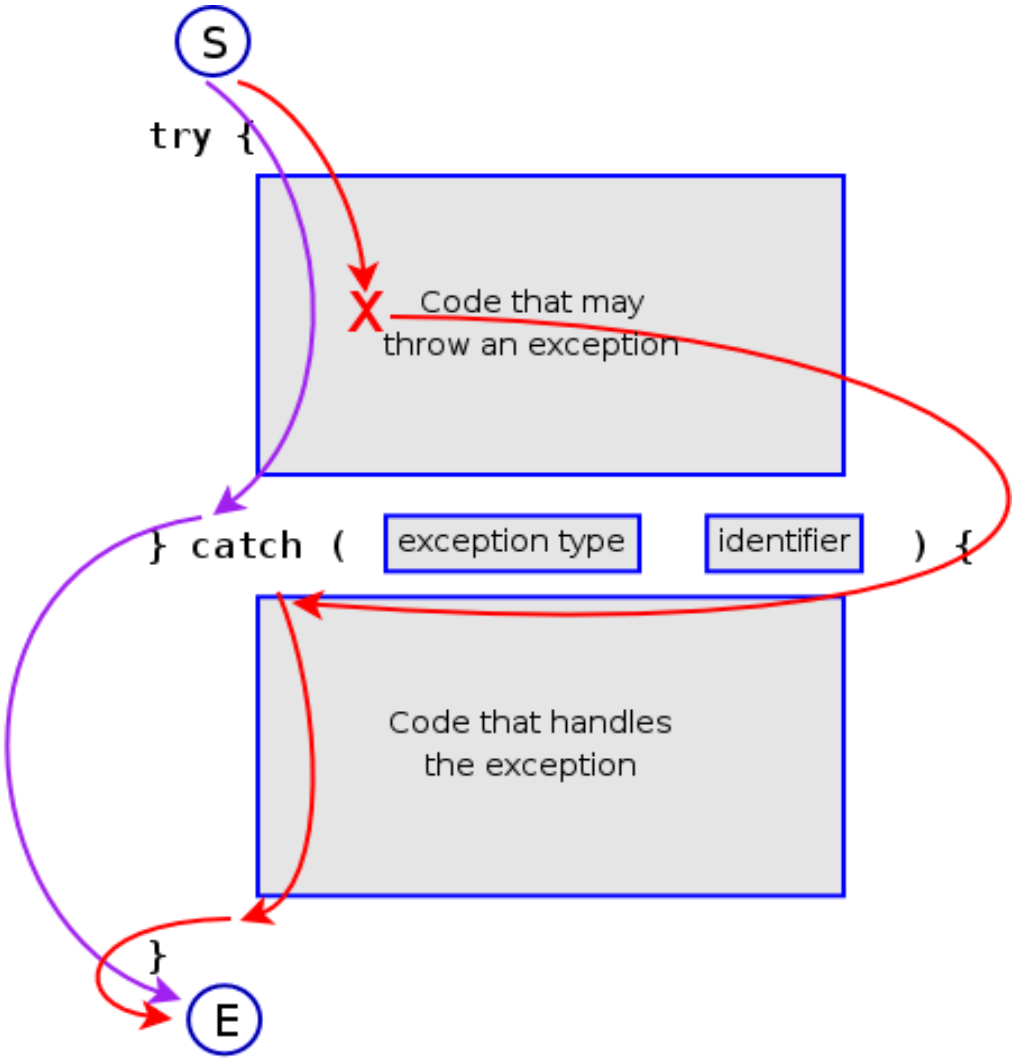
```
try{
    // The try clause is the piece of code which you want to try to execute.
   // it contains statements in which an exception could be raised
}
catch (Exception e){
   // The catch clauses are the handlers for the various exceptions.
   //it contains code to handle Exception and recover
}
```

Example of try and catch

```
try{
      myMethod();
}
catch (Exception e){
      System.err.println("Caught Exception:  " + e)
}
```

The parameter ***e*** is of type Exception and we can use it to print out what exception occurred.

Flow of control in Exception Handing



Returning to our ShopV5.0 Application

```
try {
    System.out.print("Please enter the product code: ");
    code = input.nextInt();
  }
  catch (Exception e) {
     String throwOut = input.nextLine();   //swallows
     System.out.println("Number expected - you entered  text");
   }
```

Improve – loop until input valid

```
boolean goodInput = false;
//goodInput is the Loop  Control Variable

while (! goodInput ) {
   try {
         System.out.print("Please enter the product code: ");
         code = input.nextInt();
         goodInput = true;
   }
   catch (Exception e) {
         String throwOut = input.nextLine();  //swallows
         System.out.println("Num expected - you entered  text");
   }
}
```

Using do..while

```
boolean goodInput = false;
do {
   try {
   System.out.print("Please enter the product code: ");
   code = input.nextInt();
   goodInput = true;
   }
   catch (Exception e) {
         String throwOut = input.nextLine();  //swallows
         System.out.println("Num expected - you entered  text");
   }
}  while (!goodInput);
```

Any Questions?

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/