Exception Handling

ADVANCED PROGRAMMING PRACTICES

INSTRUCTORS: DR. JOEY PAQUET

EMAIL: JOEY.PAQUET@CONCORDIA.CA

DR. AMIN RANJ BAR

EMAIL: AMIN.RANJBAR@CONCORDIA.CA

TA: PRIYANG PATEL

EMAIL: PRIYANGPATEL001@GMAIL.COM

Agenda

- What is Exception Handling?
- Advantages of Exception Handling
- Types of Java Exception
- Java Exception keywords
- Exception Handling in Java – Useful Methods
- Stack Unwinding
- Constructor Exception
- Boiler Example

What is Exception Handling

Exception Handling is a mechanism to handle runtime errors.

When an **Exception** occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.

An exception can occur for many different reasons. Following are some scenarios where an exception occurs.

- A user has entered an invalid data.
- A file that needs to be opened cannot be found.
- A network connection has been lost in the middle of communications or the JVM has run out of memory.

Advantage of Exception Handling

1. **Maintain the normal flow of the application**
2. **Separating Error-Handling Code from "Regular" Code**
3. **Propagating Errors Up the Call Stack**
4. **Grouping and Differentiating Error Types**

Types of Exceptions

1. checked Exception :

A checked exception is an exception that is checked (notified) by the compiler at compilation-time, these are also called as compile time exceptions.

2. unchecked Exception :

An unchecked exception is an exception that occurs at the time of execution. These are also called as **Runtime Exceptions**.

3. Error :

These are not exceptions at all, but problems that arise beyond the control of the user or the programmer.

Java Exception Keywords

There are 5 keywords which are used in handling exceptions in Java.

Try: The "try" keyword is used to specify a block where we should place exception code. The try block must be followed by either catch or finally. It means, we can't use try block alone.

Catch: The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.

Finally: The "finally" block is used to execute the important code of the program. It is executed whether an exception is handled or not.

Throw: The "throw" keyword is used to throw an exception.

Throws: The "throws" keyword is used to declare exceptions. It doesn't throw an exception. It specifies that there may occur an exception in the method. It is always used with method signature.

Stack Unwinding

When an exception is thrown but not caught in a particular scope, the method-call stack is "unwound," and an attempt is made to catch the exception in the next outer `try` block. This process is called **stack unwinding**. Unwinding the method-call stack means that the method in which the exception was not caught terminates, all local variables in that method go out of scope and control returns to the statement that originally invoked that method.

Constructor Exception

Constructors that throw exceptions are problematic: if a constructor fails, the object is not created, which makes it hard to recover from.

This is even more problematic with class hierarchies, which require a sequence of constructors to succeed in order for an object to be fully constructed. In Java, an object is either successfully fully constructed or it does not exist.

Exception Handling Example: Boiler

Industrial boiler controlled by software.

Connected to a pressure sensor and a pressure release valve.

Keeps the pressure within an acceptable range.

If the sensor is misbehaving, it shuts down the boiler by opening the valve.

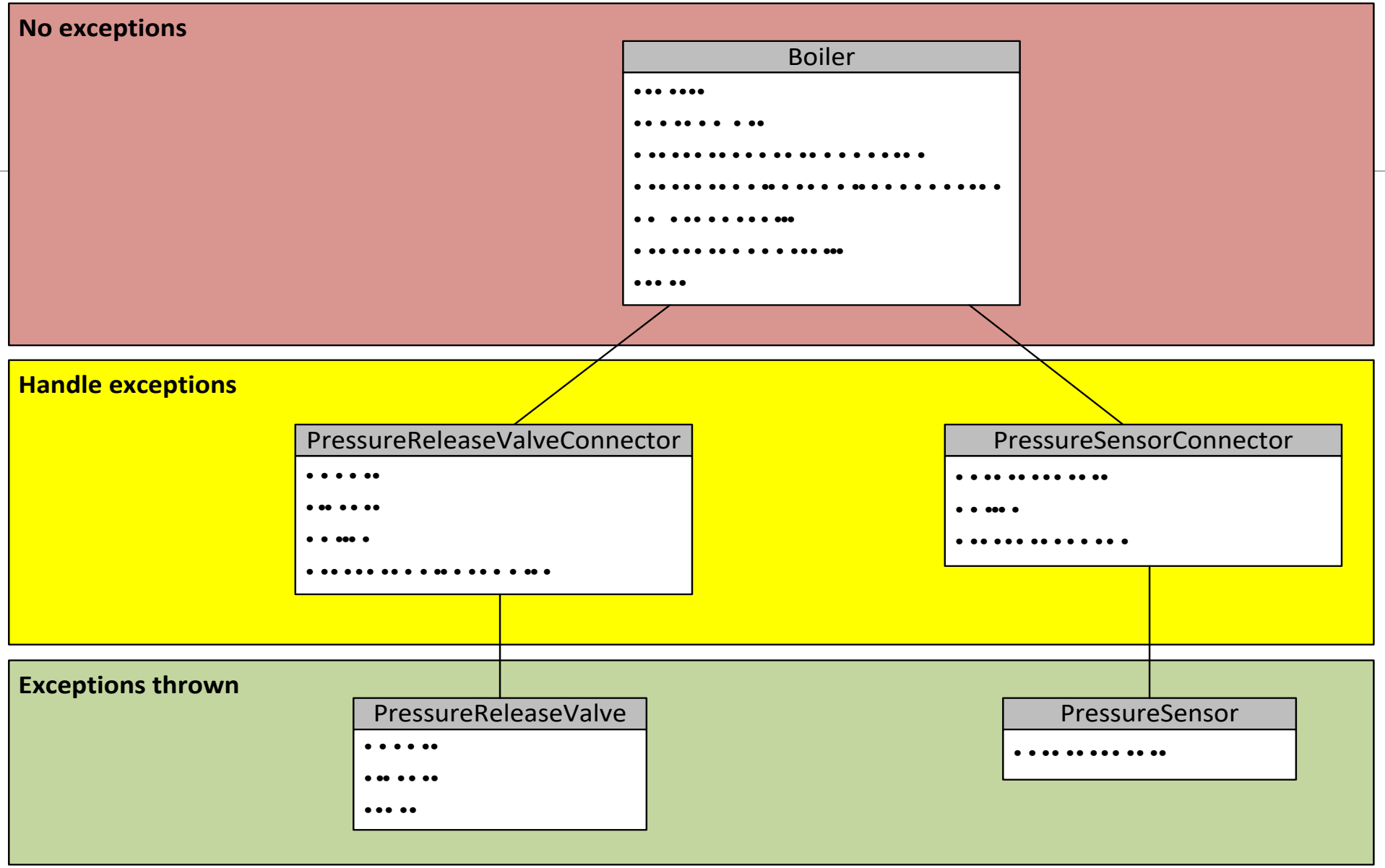
If the valve is stuck, it calls an emergency.

Keeps a log of the pressure readings, as well as another log for operational events.

Hardware drivers can throw exceptions.

For security, the boiler controller should be shielded from those exceptions.

Thus, an exception handling layer is added.


No exceptions
Boiler
Handle exceptions
PressureReleaseValveConnector
PressureSensorConnector
Exceptions thrown
PressureReleaseValve
PressureSensor

THANK YOU