Recitation 3. Debugging. Start on Exceptions

Debug perspective: Click on this icon. In pop-window, select Debug. Debug perspective opens (next slide).

Setting a breakpoint

Breakpoint: a line of program that is set so that execution will stop when it is reached during debugging.

Put mouse here, right click, select Toggle Breakpoint from contextual menu. The breakpoint is set and the dot appears.
If it was already set, it is unset.

Running the debugger

Choose Run > Debug. Execution starts but stops at the first line with a breakpoint set.

Running the debugger

Choose Run > Debug. Execution starts but stops at the first line with a breakpoint set.

Parameter and its value

Click icon Step Over to have statement executed.

Statement has been executed x appears with its value

Step into and Step over do the same thing except for executing a call.

Step over executes a call in one step.

Step into makes the first statement in body of method being called the next statement to execute, so you can see detailed execution of method body.
Recitation 3. Exceptions

public static void main(String[] args) {
    int b = 3/0;
}

Division by 0 causes an “Exception to be thrown”. program stops with output:

Exception in thread "main"
java.lang.ArithmeticException: / by zero
at C.main(C.java:7)

The "Exception" that is "thrown"

parseInt throws a NumberFormatException

public static void main(String[] args) {
    int b = Integer.parseInt("3.2");
}

Exceptions and Errors

In Java, there is a class Throwable:

- Throwable()
- Throwable(String)

An Exception is an instance of class Throwable (or one of its subclasses)

Exceptions and Errors

So many different kind of exceptions that we have to organize them.

Creating and throwing Exception

03 public class Ex {
04     public static void main(...) {
05         second();
06     }
07     second();
08     third();
09 }
10 }
11
12 ArithmeticException: by zero
13 at Ex.third(Ex.java:13)
14 at Ex.second(Ex.java:9)
15 at Ex.main(Ex.java:5)
16 }
17 at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
18 at sun.reflect.NativeMethodAccessorImpl.invoke(Native Method)
19 at sun.reflect.DelegatingMethodAccessorImpl.invoke(…)
20 at java.lang.reflect.Method.invoke(Method.java:585)
public class Ex {
    public static void main(…) {
        second();
    }
}

public class Ex {
    public static void main(…) {
        second();
    }
    public static void second() {
        third();
    }
    public static void third() {
        throw new ArithmeticException("I threw it");
    }
}

ArithmeticException: I threw it
at Ex.third(Ex.java:14)
at Ex.second(Ex.java:9)
at Ex.main(Ex.java:5)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(…)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(…)
at java.lang.reflect.Method.invoke(Method.java:585)

Won’t compile. Needs a "throws clause, see next slide
If a method throws an Exception that is not a subclass of RuntimeException, the method needs a throws clause. Don’t be concerned with this issue. Just write your method and, if Java says it needs a throws clause, put one in

The "throws" clause
/** Class to illustrate exception handling */
public class Ex {
    public static void main() throws OurException {
        second();
    }
    public static void second() throws OurException {
        third();
    }
    public static void third() throws OurException {
        throw new OurException("mine");
    }
}

If Java asks for it, insert the throws clause. Otherwise, don’t be concerned with it.

Try statement: catching a thrown exception
public class Ex1 {
    public static void main() throws MyException {
        second();
        try {
            second();
        } catch (MyException ae) {
            System.out.println("Caught MyException: " + ae);
            System.out.println("procedure first is done");
        }
        public static void second() throws MyException {
            third();
        }
        public static void third() throws MyException {
            throw new MyException("yours");
        }
    }
}

public class Ex1 {
    public static void main() throws MyException {
        second();
    }
    public static void second() throws MyException {
        third();
    }
    public static void third() throws MyException {
        throw new MyException("yours");
    }
}

/** Illustrate exception handling */
public class Ex {
    public static void main() {
        second();
    }
    public static void second() {
        third();
    }
    public static void third() {
        throw new ArithmeticException("I threw it");
    }
}

/** An instance is an exception */
public class OurException extends Exception {
    /** Constructor: an instance with message m */
    public OurException(String m) {
        super(m);
    }
    /** Constructor: an instance with no message */
    public OurException() {
        super();
    }

How to write an exception class
/** An instance is an exception */
public class OurException extends Exception {
    /** Constructor: an instance with message m */
    public OurException(String m) {
        super(m);
    }
    /** Constructor: an instance with no message */
    public OurException() {
        super();
    }

How to write an exception class
/** An instance is an exception */
public class OurException extends Exception {
    /** Constructor: an instance with message m */
    public OurException(String m) {
        super(m);
    }
    /** Constructor: an instance with no message */
    public OurException() {
        super();
    }

How to write an exception class
/** An instance is an exception */
public class OurException extends Exception {
    /** Constructor: an instance with message m */
    public OurException(String m) {
        super(m);
    }
    /** Constructor: an instance with no message */
    public OurException() {
        super();
    }

How to write an exception class
/** An instance is an exception */
public class OurException extends Exception {
    /** Constructor: an instance with message m */
    public OurException(String m) {
        super(m);
    }
    /** Constructor: an instance with no message */
    public OurException() {
        super();