Combination

- try-catch
- try-catch-finally
- try-catch-catch
- try-catch-catch-finally
- try-finally

mind the sequence:

first catch the more specific exceptions otherwise there is unreachable code

```java
package errors;

public class StartUsingOurException {
    public static void main(String[] args) {

        StartUsingOurException instance = new StartUsingOurException();

        try {
            instance.divide(2, 3);
        } catch (SoteException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            System.out.println("Program ends ");
        }
        System.out.println("Program ends ");

    }

    public int divide(int a, int b) throws SoteException {
        System.out.println("We are in myMethod");

        if (b == 0) {
            throw new SoteException("C:\\fake");
        }

        return a / b;
    }
}
```

```java
package errors;

public class StartUsingOurException {
    public static void main(String[] args) {

        StartUsingOurException instance = new StartUsingOurException();

        try {
            instance.divide(2, 3);
        } catch (SoteException e) {
            e.printStackTrace();
        } finally {
            System.out.println("Program ends ");
        }
        System.out.println("Program ends ");

    }

    public int divide(int a, int b) throws SoteException {
        System.out.println("We are in myMethod");

        if (b == 0) {
            throw new SoteException("C:\\fake");
        }

        return a / b;
    }
}
```

finally blocks are always(!) executed. Typical use case: Working with resources such as databases, networks, filesystems and the resource has to be released.

Assertions

assert is a key word in Java (means in German "zusichern"

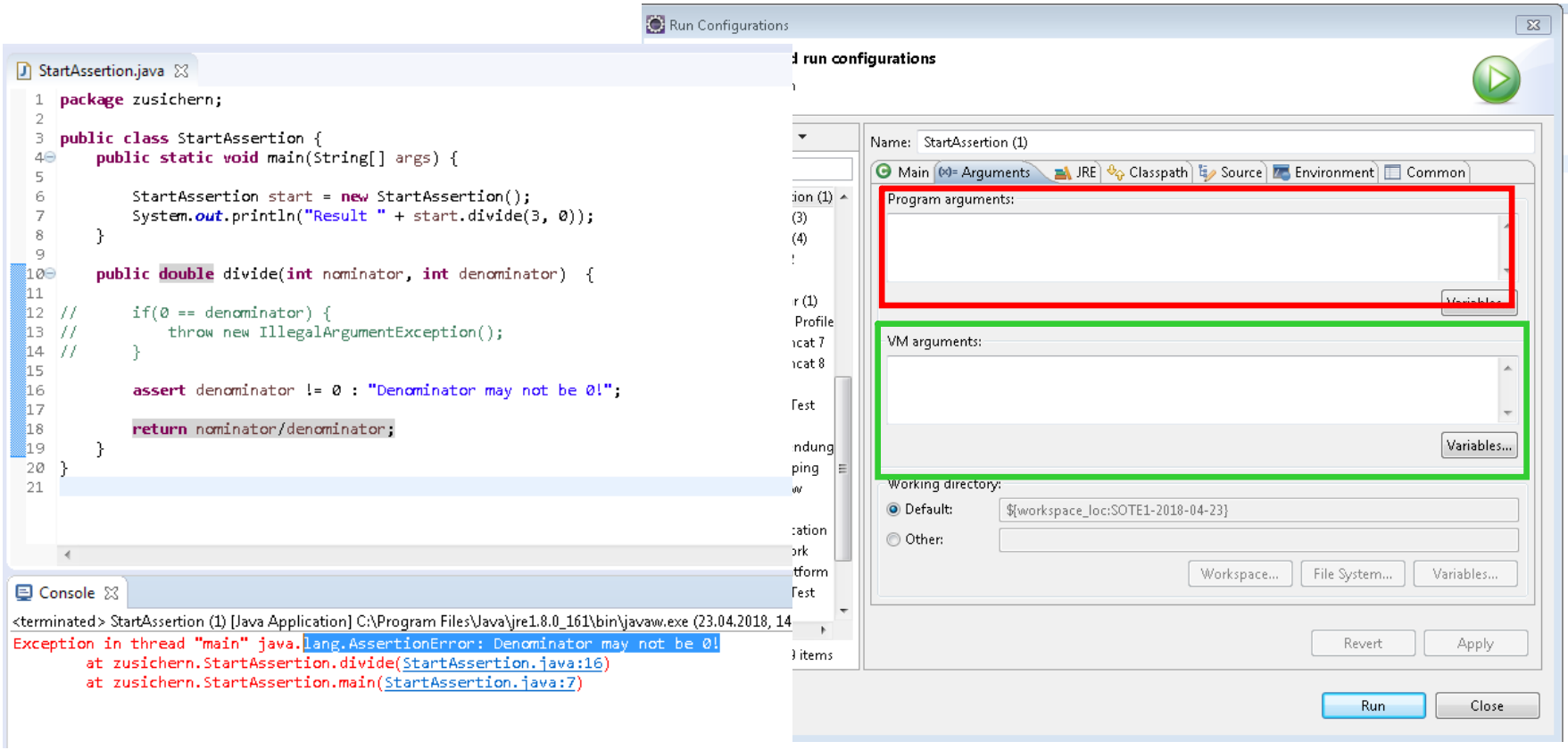
Syntax

```
assert <Condition> : <MessageIfConditionIsNotMet>;
```

Use case: Evaluate (during development time) whether pre- and post-conditions are met.

Assertion can be switched on and off with the JVM-Parameter -ea.

Recommendation: Never catch AssertionErrors



java -jvm-parameter zusichern.StartAssertion program-arguments

java -ea zusichern.StartAssertion

Reflection and Introspection

TBD next lecture

```java
public static void main(String[] args) {
    String classToBeInvoked = "reflect.Student"; //args[0];
    String methodToBeInvoked = "setName";
    String constructorParameter1 = "Justin";
    String constructorParameter2 = "007";

    try {
        //1. Select the class just based on the name (fully qualified class name -- including package)
        Class unknownClass = Class.forName(classToBeInvoked); //Student student;

        //2. Instantiate the class
        //student = new Student("Justin");
        Class[] parameterTypes = {String.class, String.class};
        Constructor constructor = unknownClass.getConstructor(parameterTypes);

        //3. Instantiate the class
        Object[] values = {constructorParameter1, constructorParameter2};
        Object instance = constructor.newInstance(values); //new Student("Justin", "007");
        System.out.println("Our Instance is " + instance);

        //4. Dump methods on class (introspection)
        Method[] methods = unknownClass.getMethods(); //public methods and inherited methods
        for(Method method :  methods) {
            System.out.println("Method " + method.getName());
        }

        System.out.println("------------");

        Method[] methods2 = unknownClass.getDeclaredMethods();//all methods declared on a class even the private one
        for(Method method :  methods2) {
            System.out.println("Method " + method.getName());
        }

        //5. select specific method
        Class[] parameterTypes2 = {String.class};
        Method method = unknownClass.getMethod("setName", parameterTypes2);

        //6. invoke the method
        Object[] params = {"Jakub"};
        Object returnValue = method.invoke(instance, params);
        System.out.println("Our Instance is " + instance + " and the return value is " + returnValue);
```

```java
public class Student {
    private String name;
    private String id;


    public Student() {
    }

    public Student(String name) {
        super();
        this.name = name;
    }

    public Student(String name, int id){

    }


    public Student(String name, String id) {
        this.name = name;
        this.id = id;
    }

    public void enroll(String date) {
        System.out.println(this.name + " is e
    }

    public String getName() {
        return name;
    }

    public String setName(String name) {
        this.name = name;
        return "Servus";
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    @Override
    public String toString() {
```