Programowanie obiektowe

Wykład 9: Wyjątki

Co to jest wyjątek?

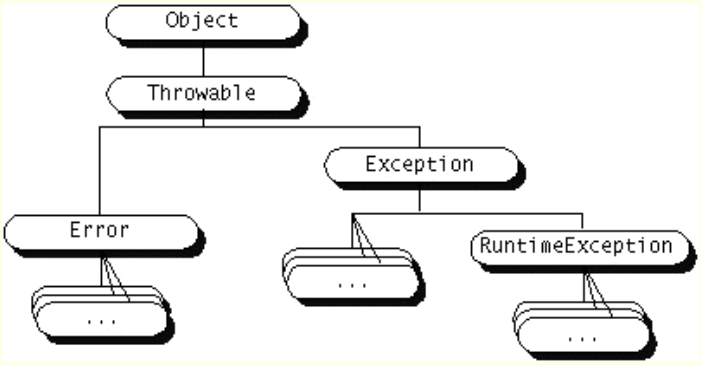
- błąd wykonania:

```
InputFile.java:8: Warning: Exception java.io.FileNotFoundException
must be caught, or it must be declared in throws clause of this method.
        in = new FileReader(filename);
```

- sytuacja wyjątkowa
- żądanie przerwania wykonywania bloku instrukcji

Wyjątek jest obiektem!

Hierarchia wyjątków



Generacja wyjątków

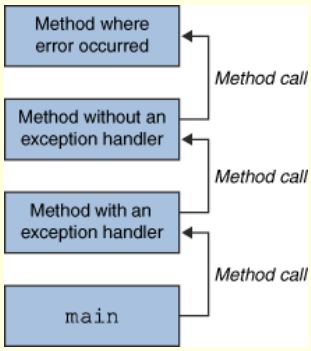
- automatyczna (np. błąd)
- programowa:

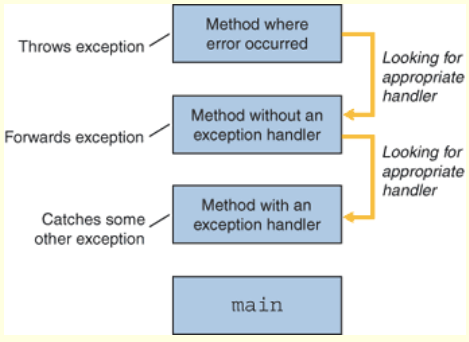
```
if (t==null)
    throw new NullPointerException("");
```

Obsługa wyjątków

```
try {
// kod który może generować wyjątki
}
catch (Typ1 wyj1) {
// obsługa wyjątków Typ1
}
catch (Typ2 wyj2) {
// obsługa wyjątków Typ2
}
catch (Typ3 wyj3) {
// obsługa wyjątków Typ3
}
finally {
// kod który wykonywany jest na końcu
}
```


Propagacja wyjątków





Tworzenie własnych wyjątków

```
class SimpleException extends Exception {}

public class SimpleExceptionDemo {
  public void f() throws SimpleException {
    System.out.println(
      ”Wyrzucam SimpleException z f()");
    throw new SimpleException ();
  }
  public static void main(String[] args) {
    SimpleExceptionDemo sed =
      new SimpleExceptionDemo();
    try {
      sed.f();
    } catch(SimpleException e) {
      System.err.println(”Złapałem!");
    }
  }
}
```

Istnieje konieczność specyfikacji wyjątków generowanych przez metody!

Konieczność ta nie dotyczy RuntimeException

Klasa Throwable

public String **getMessage**()

public String **getLocalizedMessage**()

public String **toString**()

public void **printStackTrace**()

public void **printStackTrace**(PrintStream s)

public void **printStackTrace**(PrintWriter s)

public Throwable **fillInStackTrace**()

public StackTraceElement[] **getStackTrace**()

public void **setStackTrace**(StackTraceElement[] stackTrace)

Wyjątki RuntimeException

AnnotationTypeMismatchException,
ArithmeticException,
ArrayStoreException,
BufferOverflowException,
BufferUnderflowException,
CannotRedoException,
CannotUndoException,
ClassCastException,
CMMException,
ConcurrentModificationException,
DOMException,
EmptyStackException,
EnumConstantNotPresentException,
EventException,
IllegalArgumentException,
IllegalMonitorStateException,
IllegalPathStateException,
IllegalStateException,
ImagingOpException,
IncompleteAnnotationException,
IndexOutOfBoundsException,
JMRuntimeException,
LSException,
MalformedParameterizedTypeException,
MissingResourceException,
NegativeArraySizeException,
NoSuchElementException,
NullPointerException,
ProfileDataException,
ProviderException,
RasterFormatException,
RejectedExecutionException,
SecurityException,
SystemException,
TypeNotPresentException,
UndeclaredThrowableException,
UnmodifiableSetException,
UnsupportedOperationException

Zasady specyfikacji wyjątków

- W metodach przeciążonych można zgłaszać tylko wyjątki wyspecyfikowane w klasie bazowej
- W konstruktorach podklas można dodawać nowe wyjątki

Przykład

```
class VeryImportantException extends
Exception {
  public String toString() {
    return "A very important exception!";
  }
}
class HoHumException extends Exception
{
  public String toString() {
    return "A trivial exception";
  }
}
```

```
public class LostMessage {
  void f() throws VeryImportantException {
    throw new VeryImportantException();
  }
  void dispose() throws HoHumException {
    throw new HoHumException();
  }
  public static void main(String[] args)
      throws Exception {
    LostMessage lm = new LostMessage();
    try {
      lm.f();
    } finally {
      lm.dispose();
    }
  }
}
```


Co umożliwiają wyjątki?

- naprawienie problemu i ponowne wykonanie kodu
- wydobycie się z błędu i kontunuację wykonania
- wykonanie alternatywnego kodu
- wykonanie skoków
- przekazywanie informacji o stanie pomiędzy metodami
- zakończenie programu

Pytania

1. Jakie są wymagania dotyczące wyjątków generowanych przez metody przeciążające lub przesłaniające?
2. Jakie są zasady specyfikacji generowanych wyjątków w konstruktorach?
3. Jakie są zasady obsługi wyjątków?
4. Czy wyjątek może być kolekcją?
5. Czy wyjątek można obsłużyć 2 razy?

Koniec