Programowanie komputerów

Wykład 5: „Mechanizm wyjątków w Javie”

dr inż. Walery Susłow

walery.suslow@ie.tu.koszalin.pl

Wyjątki (1)

- Java posiada zapożyczony z języka Ada mechanizm informowania o błędach: *wyjątki* (ang. Exceptions).
- Mechanizm obsługi wyjątków w Javie umożliwia obsługę sytuacji krytycznych, dzięki czemu program nie zawiesi się po wystąpieniu błędu i wykona ciąg operacji obsługujących dany wyjątek.
- Mechanizm obsługi wyjątków zwiększa niezawodność programów Java.

Wyjątki (2)

- Programista tworząc aplikację nie zawsze jest w stanie w pełni przewidzieć jej zachowanie, w szczególności gdy ma do czynienia ze zdarzeniami zależnymi od użytkownika, sieci, urządzenia zewnętrznego.
- Język Java umożliwia zabezpieczenie się przez błędami jakie mogą wystąpić podczas nieprzewidywanych zdarzeń.
- W przypadku wystąpienia błędu *sterowanie* programu *wychodzi z aktualnie wykonywanej metody* i przechodzi do specjalnego bloku obsługi wyjątku.

Wyjątki (3)

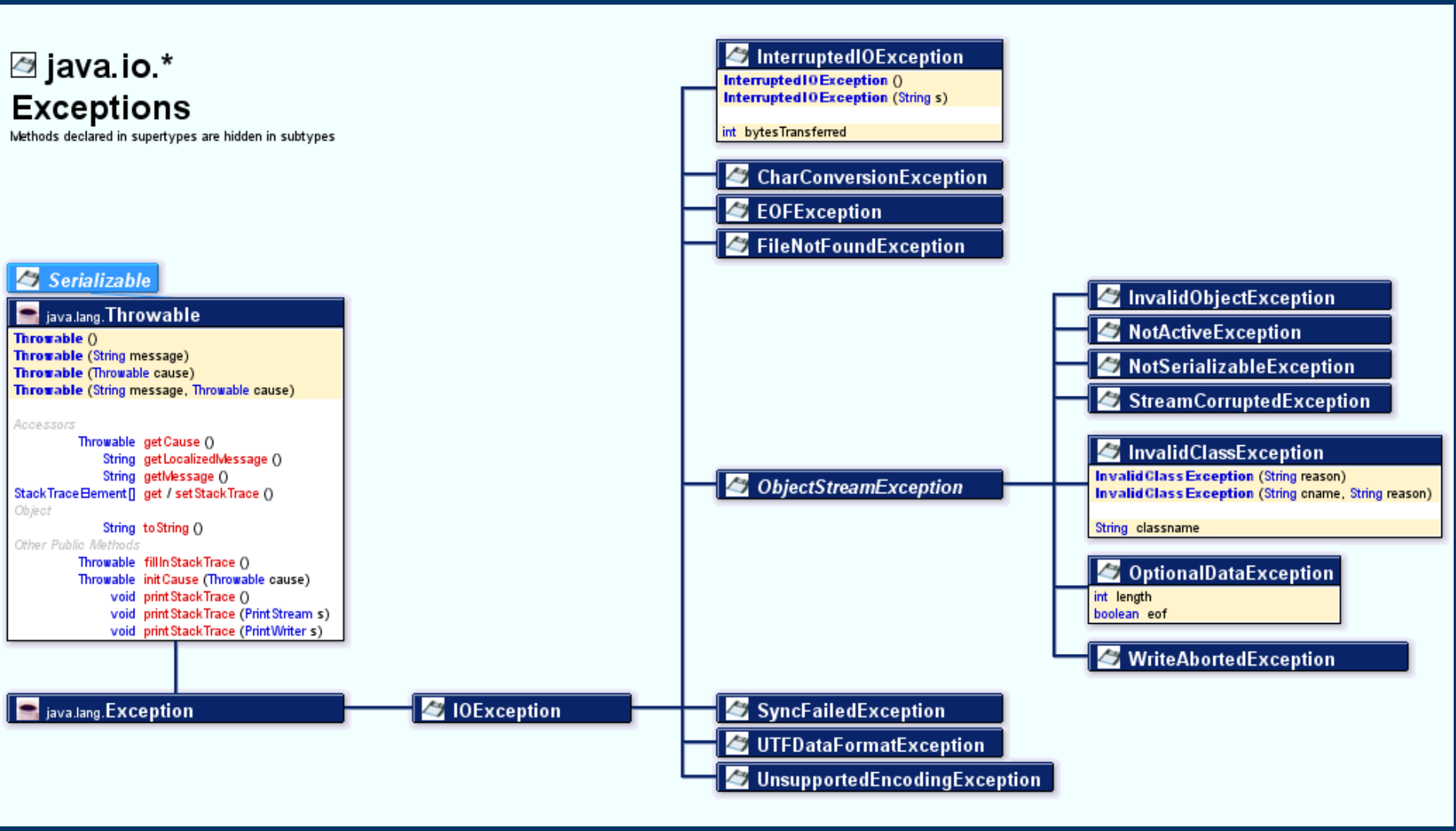
- *wyjątki kontrolowane (ang. checked exeptions)* – stanowią większość wyjątków w Javie. Znaczy to, że kompilator jest w stanie sprawić, że każda metoda zgłasza tylko te wyjątki, które są jawnie wymienione w jej deklaracji.
- *wyjątki niekontrolowane (ang. unchecked exeptions)* – standardowe wyjątki i błędy zgłaszane przez system wykonawczy, które są podklasami klas *RuntimeExeption* i *Error*.

Kiedy obsłużyć wyjątek

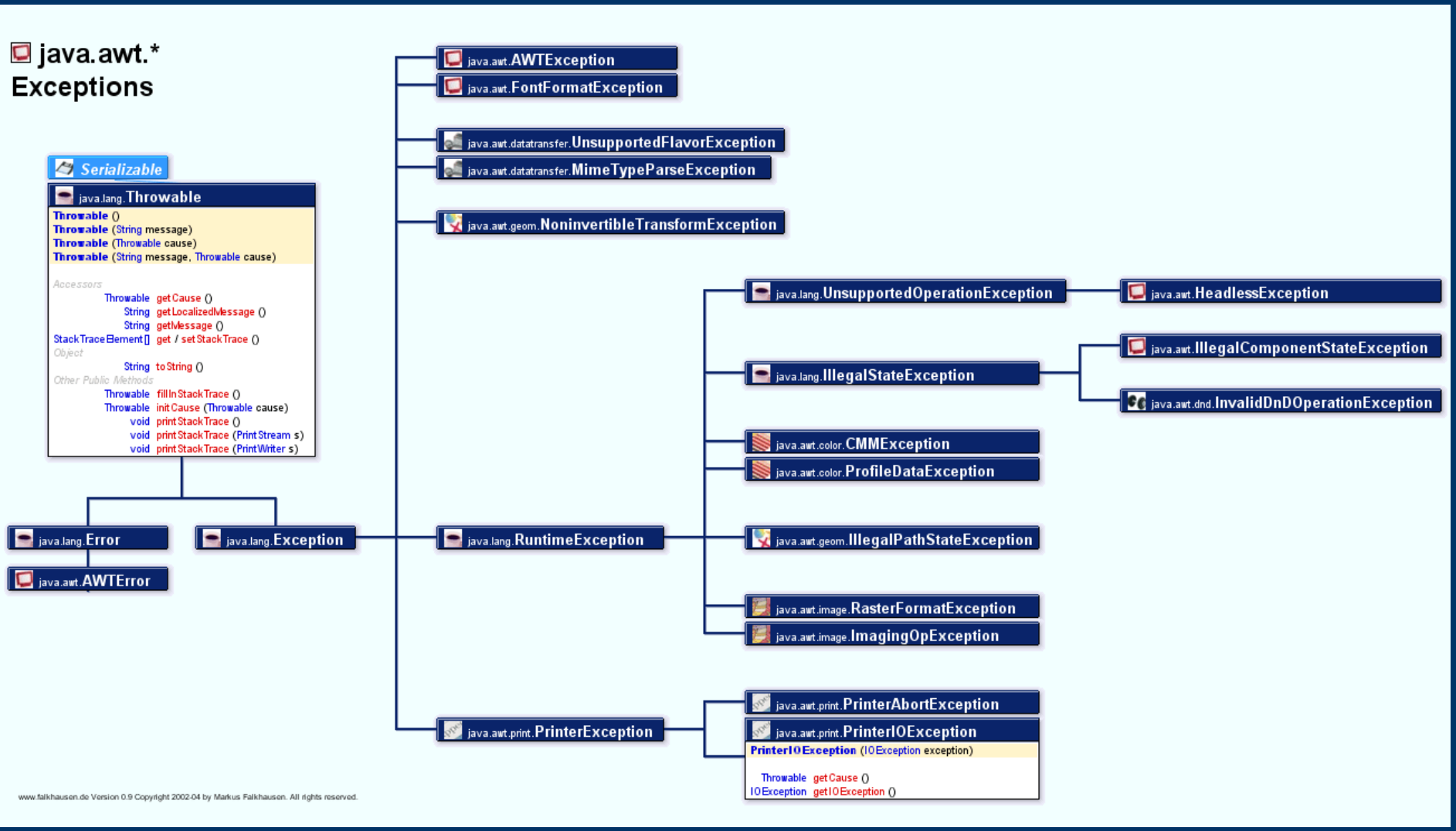
- Programista sam może decydować czy w danym momencie ma być „wyrzucany” wyjątek czy nie.

- Zasada: wyjątek jest wtedy gdy w danym momencie przetwarzania danych nie mamy wystarczająco wiedzy aby poradzić sobie z powstałym problemem.

Klasy Java do obsługi wyjątków (1)



Klasy Java do obsługi wyjątków (2)



Użyteczne metody klasy Throwable

- *String getMessage()* - zwraca napis podany kontruktorowi wyjątku Exception.
- *String toString()* - zwraca napis reprezentujący nazwę wyjątku (klasy wyjątku).
- *void printStackTrace()* - wypisuje *call stack trace* czyli sekwencję metod, która została wywołana do momentu wyrzucenia wyjątku.

Główne klasy obsługi wyjątków

- Wyjątki typu *Error* występują wtedy, gdy wystąpi sytuacja specjalna w maszynie wirtualnej. Nie powinny one być obsługiwane w "zwykłych" programach Java.
- W "zwykłych" programach generowane są i obsługiwane obiekty, które dziedziczą z klasy *Exception*. Wyjątek tego typu oznacza, że w programie wystąpił błąd, lecz nie jest to poważny błąd systemowy.
- *RunTimeExceptions jest to* szczególna podklasa klasy Exception. Obsługuje ona wyjątki, które występują podczas wykonywania programu. Występują one np. wtedy, gdy zostaną wyczerpane zasoby systemowe lub nastąpi odwołanie do nie istniejącego elementu tablicy lub pliku.

Deklarowanie typów wyjątków

Do opisu sytuacji wyjątkowych wygodnie jest mieć więcej danych niż tylko napis, który udostępnia klasa *Exception*. W razie potrzeby, możemy utworzyć podklasę klasy *Exception*, która będzie zawierała niezbędne dodatkowe dane.

Przeanalizujmy sytuację dot. aplikacji „Egzamin”, w której będziemy potrzebowali zgłosić przykładowy wyjątek *NieZdalEgzaminu*.

Zgłaszany wyjątek powinien zawierać informację: kto nie zdał egzaminu i z jakiego przedmiotu.

```
public class NieZdalEgzaminu extends Exeption{
	public Student student;
	public String przedmiot;

	NieZdalEgzaminu(Student nieprzygotowany, String
nazwaPrzedmiotu){
		super("Student " + nieprzygotowany.nazwisko+ "nie
zdal " + przedm);
		student=nieprzygotowany;
		przedmiot=nazwaPrzedmiotu;
	}
}
```

Wykorzystanie instrukcji throw i klauzuli throws (1)

Instrukcja *throw* – służy do zgłaszania wyjątków. Wymaga podania obiektu reprezentującego wątek.

Klauzula *throws* – w jej treści opisywane są wyjątki kontrolowane przez daną metodę (lista nazw oddzielonych przecinkami).

```
nazwaMetody(zmienne) throws Wyjątek1, Wyjątek2
{       instrukcje;
        ...
        throw Wyjatek1;
        ...
}
```

Wykorzystanie instrukcji throw i klauzuli throws (2)

```
public void przeprowadzEgzamin(Student student, String
przedmiot) throws NieZdalEgzaminu
{       ...

        //instrukcje

        ...
        if(student.ocena==2) throw NieZdalEgzaminu(student,
przedmiot);
        else student.wpiszDoIndexu(przedmiot, student.ocena);
}
```

Wykorzystanie instrukcji try oraz klauzul catch i finally (1)

Wyjątki można wychwytywać za pomocą instrukcji *try*. Składnia bloku *try* jest następująca:

```
try
        //instrukcje
catch(typ_wyjatku id)
        //instrukcje
catch(typ_wyjatku id)
        //instrukcje
        ...
finally
        //instrukcje
```

Wykorzystanie instrukcji try oraz klauzul catch i finally (2)

- Instrukcje umieszczone w sekcji *try* są wykonywane do chwili zgłoszenia wyjątku.
- Jeśli zostanie zgłoszony wyjątek, to będą przeszukane kolejne klauzule *catch* w celu znalezienia klasy (lub superklasy) zgłoszonego wyjątku.
- Jeżeli klasy wyjątku nie uda się znaleźć, to wyjątek jest propagowany na zewnątrz instrukcji *try*.

- Fakultatywna klauzula *finally* będzie zawsze wykonana po wszystkich czynnościach związanych z obsługą klauzul catch.

Wykorzystanie instrukcji try oraz klauzul catch i finally (3)

```
try{
        przeprowadzEgzamin(ProgramowanieJava);
}
catch(NieZdalEgzaminu e){
        System.out.println("Niestety, tym razem nie udało się.");
}
finally{
        System.out.println("Dziękuję za interesującą wymianę
zdań!");
}
```

Wykorzystanie instrukcji try oraz klauzul catch i finally (4)

```
//inny przykład, dot. obsługi wejścia/wyjścia
try{
  BufferedReader in = new BufferedReader(
  new FileReader(args[0]));
  String s;
  while((s=in.readLine())!=null)
  System.out.println(s);
  in.close();
}catch(FileNotFoundException e){
  System.err.println(e);
//Zwróć uwagę na miejsce docelowe komunikatu System.err
}catch(IOException e){
  System.err.println(e);
}
```

Tworzenie własnych wyjątków

```
class Lapsus extends Exception{...}

public class Test{
  public void zaplanowaneDzialania() throws Lapsus{
  //spodziewamy się powodzenia, ale...
  throw new Lapsus();
  }

  public static void main(String[] args){
  Test t = new Test();
    try{ t.zaplanowaneDzialania();
        } catch(Lapsus e){
                  System.err.println(„Przechwycony błąd!”);
                  e.printStackTrace();
                  }
  } }
```

Kiedy używać wyjątków?

- Wybór sytuacji, w których należy zgłosić wyjątek, nie podlega żadnym sztywnym regułom.
- Należy jednak pamiętać o tym, aby nie nadużywać wyjątków jako metody zgłaszania sytuacji normalnych i oczekiwanych.