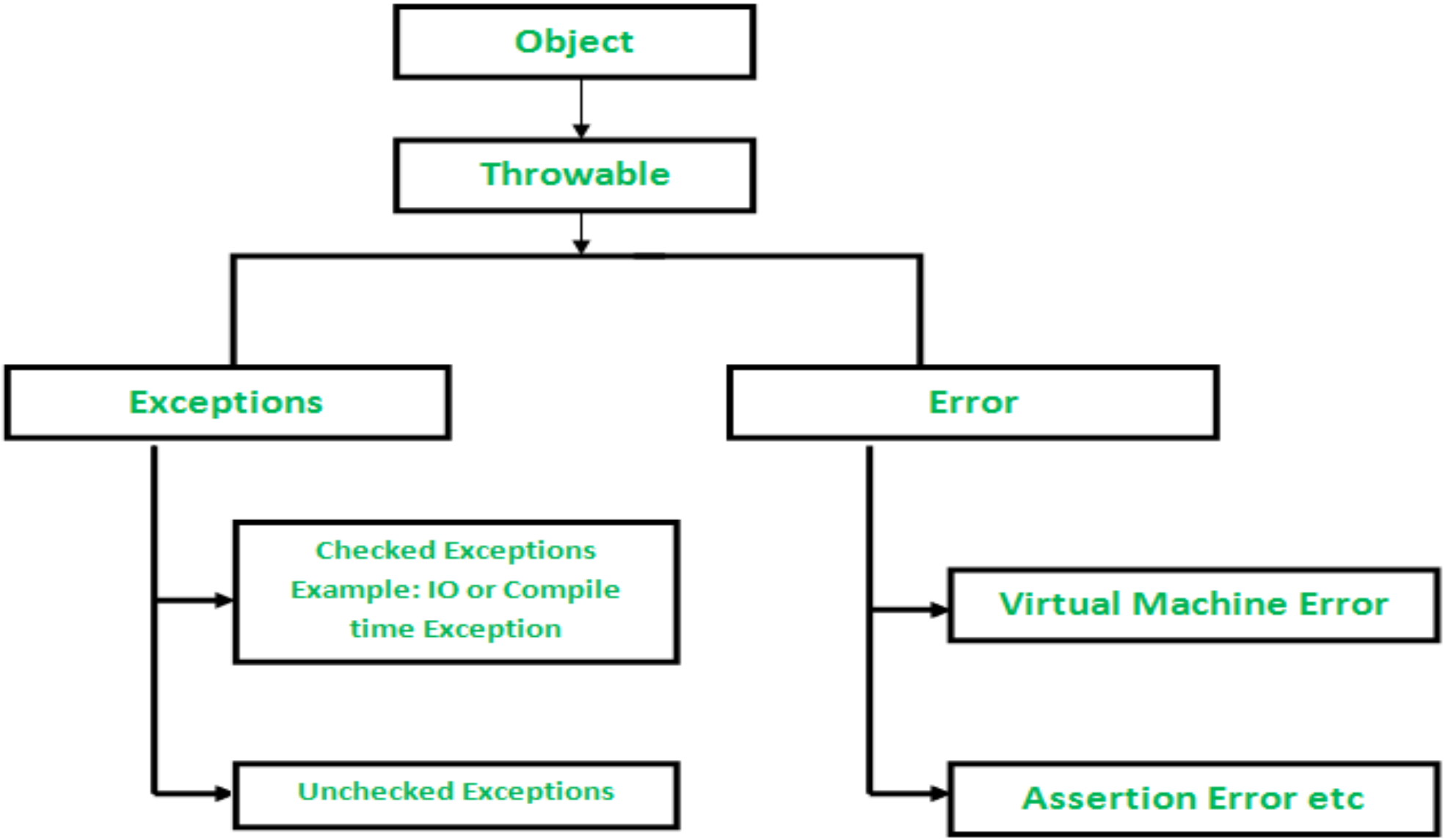
Daniel Górski
Korporacyjna Java
Wykład 7: Wyjątki

Wyjątki w Javie



Throwable

- Nadklasa wszystkich wyjątków
- Wprowadzona w Java 1.4 (2002 rok)
- Zmiany w klasie Throwable mogą wymagać zmian w JVM
  - Np dodanie nowego pola inicjowanego z wartością non-null
- Ponad 1000 linii kodu

Error

- Błąd na tyle duży, że aplikacja nie powinna próbować łapać go i kontynuować działania
- Rzucany w celach diagnostycznych
  - OutOfMemoryError
  - StackOverflowError
  - AssertionError
    - Słówko kluczowe assert
    - Domyślnie wyłączone rzucanie błędu
    - Opcja kompilacji -ea / -enableassertions

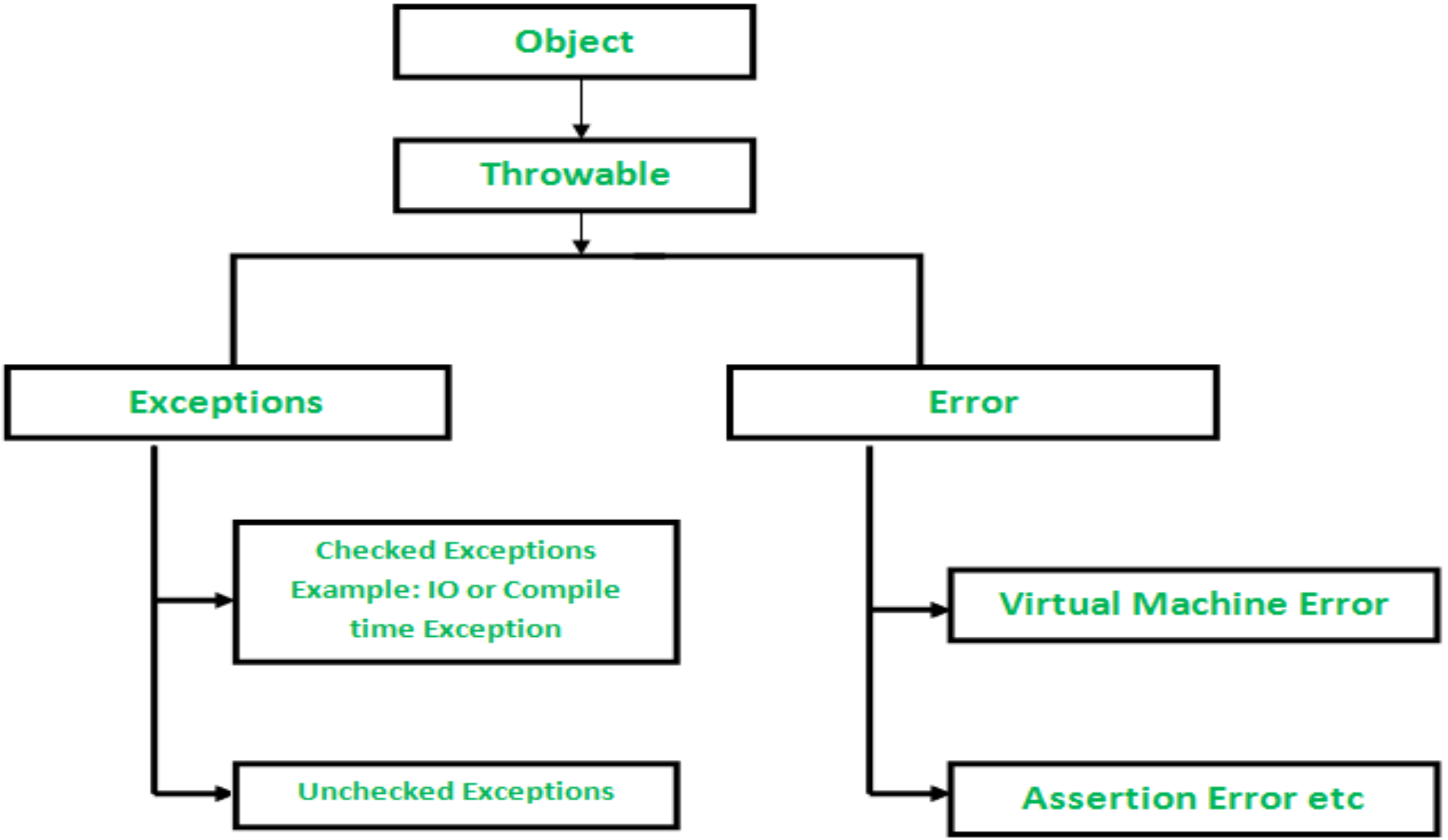
Exception

- Wyjątek oznaczony
- Rozszerza Throwable i sam w sobie nie ma istotnego nowego kodu
  - IOException
  - NamingException
  - ParseException
  - ReflectiveOperationException
    - ClassNotFoundException
    - IllegalAccessException
    - NoSuchFieldException

RuntimeException

- Wyjątek nieoznaczony
- Rozszerza Exception i sam w sobie nie ma istotnego nowego kodu
  - NullPointerException
  - ArithmeticException
  - ClassCastException
  - IllegalArgumentException
  - IllegalStateException
  - UnsupportedOperationException
  - NoSuchMechanismException

Wyjątki w Javie



Blok try-catch-finally

- Możliwości:
  - try-catch(? extends Throwable)-finally
  - try-catch(? extends Throwable)
  - try-finally
- Blok try wykonuje się pierwszy
- Blok catch wykona się tylko gdy blok try zostanie przerwany "łapanym" typem wyjątku
- Blok finally wykona się jako ostatni niezależnie od tego czy w sekcji try będzie wyjątek czy nie, chyba, że...

Kiedy blok finally się nie wykona

- Gdy w blokach try / catch wykona się:
  - System.exit(exitStatus)
  - Runtime.getRuntime().halt(exitStatus)
  - Nieskończona pętla
  - Blokująca operacja, która nie wykona się nigdy
  - JVM przestanie działać
    - Zewnętrzna przyczyna: zabicie procesu / problemy sprzętowe
    - Wewnętrzna przyczna: brak pamięci / zbyt długa rekurencja

finally i return

- W bloku try może być polecenie return
  - Blok finally i tak się wykonuje przed powrotem kontroli poziom wyżej
  - Jeśli w bloku try zwracana była zmienna Immutable to przypisanie nowej wartości do tej zmiennej w bloku finally nie powoduje zmiany wartości zwracanej
  - Jeśli w bloku finally nastąpią zmiany obiektu Mutable to naturalnie zmiany zachodzą
- Jeśli w bloku finally pojawi się return to nadpisuje return z bloku try i catch

Teoria: wyjątek oznaczony vs nieoznaczony

- Wyjątek oznaczony jest wyjątkiem przewidywalnym
  - Metoda musi informować o tym, że może go rzucić
- Wyjątek nieoznaczony jest wyjątkiem "nieprzewidywalnym"
  - Metoda nie musi informować o tym, że może go rzucić
- Obydwa mogą być "skonsumowane" przez blok try / catch
  - Error w teorii również może być skonsumowany, ale nie jest to mile widziane

Wyjątek oznaczony vs nieoznaczony: po co Nam takie rozróżnienie

- W teorii:
  - Potencjalnie każde dzielenie liczb może spowodować ArithmeticException
  - Potencjalnie każde odwołanie się do elementu tablicy może spowodować IndexOutOfBoundsException
  - ...
  - Potencjalnie każde odwołanie do zmiennej obiektowej może spowodować NullPointerException
    - Powstał nowy język programowania: Kotlin który ma zapobiegać temu wyjątkowi

Kotlin: jedna ważna cecha w wielkim skrócie

- Kotlin jest językiem opartym na JVM
- Typy Nullable i Non-nullable
  - np String? i String
- Na poziomie kompilacji jest to stwierdzane
  - Np mamy Map<Integer, String> simpleMap
  - simpleMap.get(5) zwraca obiekt typu String?
    - Bo nie wiadomo czy taki obiekt w mapie jest
  - simpleMap.getOrDefault(5, "") zwraca obiekt typu String
    - Mamy pewność, że to nie null

Praktyka: wyjątek oznaczony vs nieoznaczony

- Zdarzają się regularnie klasy typu BusinessException extends Exception
  - Te wyjątki są zwykle łapane w blokach try-catch na wyższym poziomie
- Również regularnie zdarzają się klasy typu TechnicalException extends RuntimeException
  - Czasem są łapane w blokach try-catch na wyższym poziomie
  - Mogą być traktowane inaczej na poziomie konfiguracji logowania (aby być wyraźnie widoczne w statystykach)

Praktyka...

- Użytkownikowi końcowemu nigdy nie powinien wyświetlić się błąd techniczny
  - Jest to nieprofesjonalne
  - Może zdradzić istotne elementy implementacji systemu
- Standardem jest konsumowanie wszystkich wyjątków na wysokim poziomie
  - Do użytkownika idzie komunikat biznesowy o błędzie: mniej lub bardziej precyzyjny
  - Do Naszych logów idzie log techniczny ze zdarzenia

Praktyka w aplikacjach Web'owych

- Jest wsparcie framework'ów do generycznej obsługi błędów
- Standardem jest rozpoznawanie najczęściej występujących przypadków
- Generyczny błąd dla nieokreślonych
  - Zwykle cykliczne monitorowanie pojawiających się błędów i ewentualne podejmowanie akcji

Pracownia: co wygląda dobrze

- Testy kilku przypadków dla każdej figury
  - Bardzo przydatne
- Automatyczna rejestracja poleceń jako wszystkich istniejących klas danego typu
  - W Springu jest to w standardzie

Pracownia: co wygląda na ciężkie w utrzymaniu / niełądnie stylowo

- Logika wypisywania w wielu miejsach
  - Można się zastanowić nad zunifikowanym wypisywaniem figury w nadklasie: abstrakcyjna figura może mieć metody abstrakcyjne jak nazwa i mapa właściwości z nazwami i na tym pracować
  - Już niedługo dojdzie wymaganie np z formatem liczb zmiennoprzecinkowych

Pracownia: co można wprowadzić, aby kod dobrze rozszerzalny

- Ponieważ:
  - Figura zna swoją nazwę (może to być nazwa klasy)
  - Figura wie jakie ma właściwości
  - Figura wie ile właściwości potrzebuje do obliczeń
- Na wysokim poziomie możemy pracować tylko i wyłącznie na abstrakcji:
  - Zarządca ma tylko zdefiniowaną listę przykładowych instancji figur / fabryk produkujących te figury
  - Zarządca wie dzięki temu jakie figury może dać do wprowadzenia

Pracownia: co można wprowadzić, aby kod dobrze rozszerzalny

- Zarządca wie jakie właściwości ma każdy typ figur i wie ile tych właściwości musi zebrać
- Zarządca może utworzyć nową figurę danego typu tylko za pomocą mapy właściwości z wartościami

- Konkretna figura sama zarządza swoimi wyjątkami, poprawnością wejścia, obliczeniami
  - Wspólne akcje oczywiście implementujemy w nadklasie
- Zarządca nie zna szczegółów implementacyjnych figur i dodanie nowej figury jest praktycznie przezroczyste z jego perspektywy

Pracownia

- Wypisywana lista figur ma być numerowana
- Program ma umożliwić rozwiązywanie podstawowych zadań geometrycznych dla koła
- Dochodzi nowa akcja: daj mi koło opisane na wybranej figurze (i dodaj je do listy wprowadzonych figur)

Koło

- Charakterystyka figury:
  - Promień
  - Pole powierzchni
  - Obwód
- Możliwe wejście: (1 cecha)
  - Promień
  - Pole powierzchni
  - Obwód