



Systemy rozproszone

Informatyka, sem. 6 część II



Zasady poprawnej obsługi wyjątków

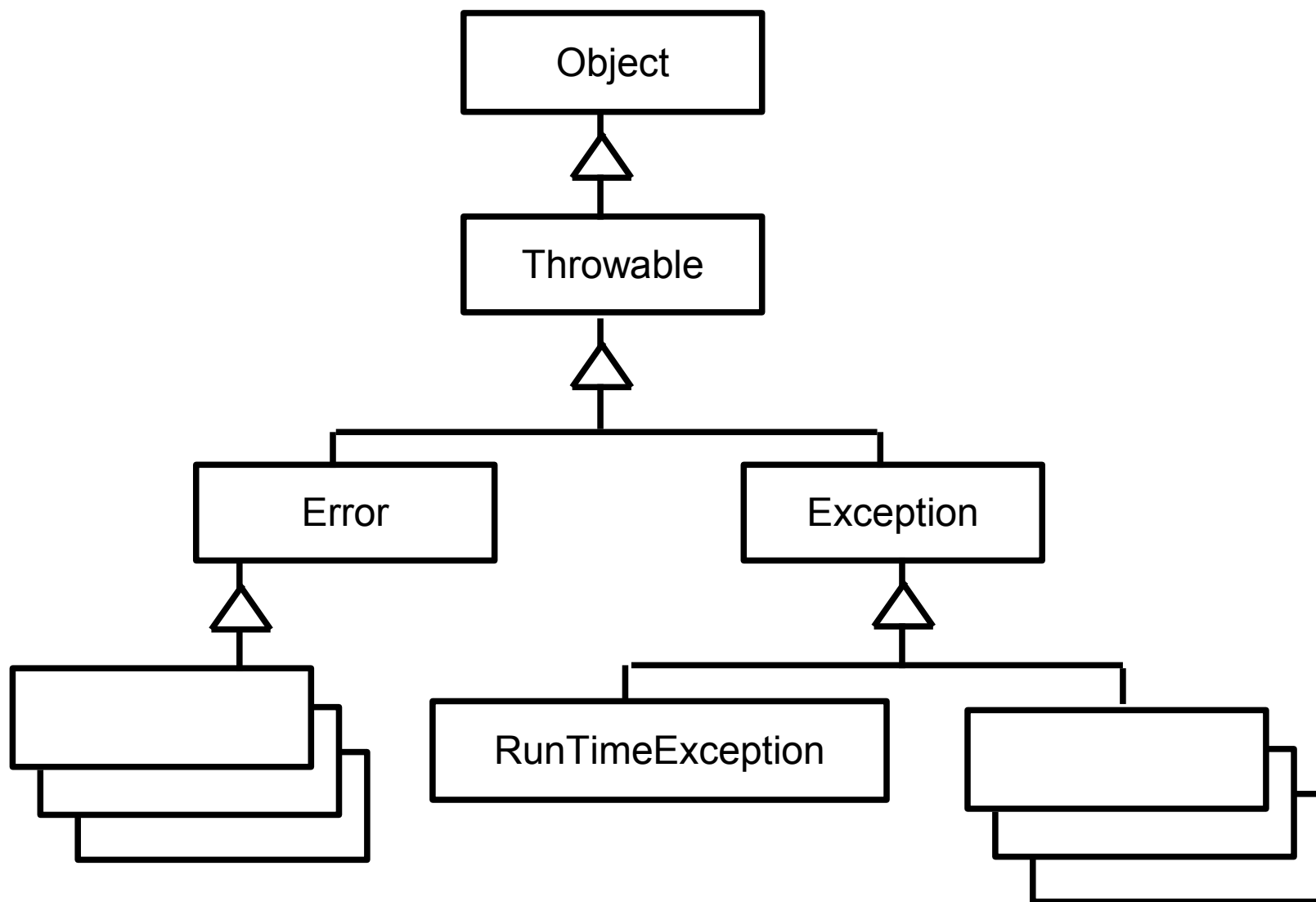


Wykorzystanie wyjątków i kontrola błędów

- Od 60% do 80% kodu programów to kod obsługi błędów i wyjątków
 - 80% z tego kodu jest nie przetestowane lub jest uruchamiane bardzo rzadko
- Reakcja na błąd
 - zwrócenie kodu błędu
 - zwrócenie wyjątku
- Wyjątek, funkcja obsługi, bloki chronione



Wyjątki w Javie





Wykorzystanie wyjątków do sytuacji wyjątkowych

- Nie należy używać wyjątków w zwykłych sytuacjach
 - optymalizacja JVM
 - zaciemnienie kodu, możliwości błędów
- Dobre API nie powinno zmuszać klienta do używania wyjątków w zwykłych sytuacjach
- Nie tak:
 - `try { int i;`
 - `while (true) a[i++].f();`
 - `} catch (ArrayIndexOutOfBoundsException e) {...}`



Wyjątki typu Runtime oraz checked

- Check exceptions - oczekujemy, że wywołujący będzie w stanie się z nich wydostać
 - przekazanie informacji pomocnej do naprawy
- Runtime (error) - wskazują na błąd w programie
 - "Unchecked" wyjątki powinny dziedziczyć po RuntimeException
 - Zwyczaj wskazuje, że Error jest zarezerwowany dla błędów JVM
- Nie zawsze można określić jednoznacznie
 - np. brak zasobów wystarczających



Zakres wykorzystania wyjątków



- Nie należy nadmiernie wykorzystywać wyjątków "checked"
 - wymagają od programisty uwagi
 - jeżeli klient nic nie może zrobić, to lepszy jest Runtime
 - alternatywa: funkcja sprawdzająca
- Nie należy wyrzucać wyjątków zbyt ogólnych ani zbyt szczegółowych.



Użycie standardowych wyjątków



- Code (exception) reuse - highly recommended
 - your API is easier to use
 - more clear application
- Common exceptions for reuse
 - `IllegalArgumentException`
 - `IllegalStateException`
 - but throw `NullPointerException` if a null value has been passed rather than an object, the same for `IndexOutOfBoundsException`
 - `ConcurrentModificationException`
 - `ArithmeticException`



Propagacja wyjątków / poziom abstrakcji

- Podczas propagacji informacja o wyjątkach nie może być gubiona
- Propagacja między warstwami powinna być minimalna
- Wyższe warstwy powinny przechwytywać niższe wyjątki i wyrzucać odpowiednie do poziomu abstrakcji
 - Exception wrapping - opakowanie wyjątku bazowego innym
 - np. SQL syntax exception, Transfer failure exception



Śledzenie wyjątków

- Javadoc powinien definiować wyjątki i sytuacje wyrzucenia (@throws)
- Wyjątków unchecked nie podaje się w definicji metody
- Wypisanie informacji o wyjątku powinno zawierać informacje o danych, które go spowodowały, np. przekroczony zakres



Atomowość błędu

- Zachowanie spójności systemu
 - kontekst wykonania, uruchomienie destruktorów
 - błędnie wywołana metoda powinna pozostawić obiekt w stanie, w jakim był przed wywołaniem
- Sposoby
 - obiekty stałe
 - sprawdzanie poprawności danych wejściowych
 - naprawianie stanu ze stanu bieżącego
 - kopie zapasowe
- Nie przechwytywać wyjątków dla przechwycenia (catch (Exception e))



Obsługa wyjątków podczas wytwarzania

- Projekt aplikacji powinien iść w parze z projektem obsługi wyjątków (zazwyczaj nie idzie)
- Eksponowane problemy w czasie wytwarzania aplikacji
 - ułatwia testowanie i wytwarzanie, uniemożliwia pominięcie wyjątku
 - ukrywanie w ostatecznej wersji
- Połączenie (mądre) obsługi wyjątków i logowania
 - logowanie / ignorowanie wyjątków w serwerze / kliencie
 - np. klient dzwoni do serwisu technicznego i podaje, że ...
 - nie logujemy wyjątków, jeżeli propagują



Zalety obsługi wyjątków

- Zalety obsługi wyjątków
 - rozdzielenie zwykłego i wyjątkowego wykonania
 - jasna specyfikacja programu
 - konieczność reakcji na wyjątek
 - łatwe wykonanie podstawowego modelu obsługi



Częste niejasności

- Wyjątki nie zmniejszają ilości kodu do napisania
 - dodatkowe funkcje naprawcze (80% kodu)
- Wyjątki nie służą do kontroli przepływu sterowania
 - mechanizm “wyjątkowy”
 - brak optymalizacji kompilatorów
 - trudności w śledzeniu i skomplikowanie kodu (wiele wyjść)
- Kody błędów nie są równoważne wyjątkom



Literatura

- J. Bloch: "Effective Java. Programming Language Guide"
- Arthur Andersen Exceptions in a J2EE Environment - A practical approach
- S. Shenoy, IBM DeveloperWorks "Best practices in EJB exception handling"
- B. McLaughlin, IBM developerWorks "EJB best practices: Build a better exception-handling framework"
- <http://wiki.org> - "Exception handling patterns"