Java – podstawy języka

Wykład 2

Klasy abstrakcyjne, Interfejsy, Klasy wewnętrzne, Anonimowe klasy wewnętrzne.

Wyjątki: obsługa błędów

Dziedziczenie cd.

- Dziedziczenie
  - Zachowanie struktury
  - Przeładowanie metod
    - » ostrożnie z metodami prywatnymi
  - Klasa ma TYLKO jedną Klasę przodka

Dziedziczenie cd.

- Występujące instrukcje
  - extends
  - super
  - final
    - pola typu prostego -> stała
    - pola obiekty -> stała referencja
    - metoda -> zakaz przeładowania
      - » final <-> private
    - klasa -> zakaz dziedziczenia

Dziedziczenie cd.

- Inicjowanie klasy potomnej
  - pola i konstruktor klasy przodka
    - Jeżeli przodek dziedziczy odpowiednio rekurencyjnie
  - pola i konstruktor klasy potomnej

Przykład 2_1

Abstrakcyjne klasy/metody

- Abstrakcyjna klasa
  - » abstract class Nazwa
    - Nie można zainicjować obiektu
    - Tylko dziedziczenie
- Abstrakcyjna metoda
  - » abstract public typ nazwa(argumenty)
    - Sama definicja – brak deklaracji
    - Wymuszona deklaracja przy dziedziczeniu

Interfejsy

- Tworzenie standardu
  - Same deklaracje brak definicji
  - Nie można zainicjować obiektu tego typu
  - Wszystkie pola static i final (niejawnie)
  - Dziedziczenie interfejsów -> interfejs
  - Implementacja -> definicja metod
  - Klasa może implementować wiele interfejsów
- Dodatkowe zastosowanie
  - Zbiór stałych np.

```
public interface  Miesiace
{
    int  STYCZEŃ=1, LUTY=2,
         MARZEC=3, … GRUDZIEŃ=12;
}
```

```
interface Pracownik {
    public String toString();
    public String stanowisko();
    public int pensja();
}

class Wykładowca implements Pracownik
{   private String imie; ....
    public String toString(){...}
    public String stanowisko(){...}
    public int pensja() {...}
    public int iloscZajec() {...}
}

class Techniczny implements Pracownik
{   private String imie; ....
    public String toString(){...}
    public String stanowisko(){...}
    public int pensja() {...}
}
```

```
    ....
danePracownika (Pracownik osoba)
 {  System.out.print(osoba);
    System.out.println(osoba.stanowisko() );
    System.out.println(osoba. iloscZajec() );
}

    ....
```

Polimorfizm

- Obiekt klasy potomnej
  - Jest typu klasy przodka
    - Posiada wszystkie jej metody
  - Można ograniczyć „w górę”
    - Zapominanie typu podstawowego
      - » Można odzyskać tę informację

Przykład 2_2

Polimorfizm

- Tablica o elementach dowolnych

```
Object[ ] tab = new Object[3];
tab[0]=”Traktor”;              //Klasa String
tab[1]=new Integer(13); //Klasa Integer
tab[2]=new Punkt();      //Klasa Punkt
…..
class Punkt
{ public int x,y;}
```

- Niestety ograniczenie typu do Object
  - niemożna wykonać operacji tab[2].x=12;

Klasy wewnętrzne

- Klasa definiowana wewnątrz innej
  - Grupowanie logiczne typów
  - Zwiększenie bezpieczeństwa
    - klasa zewnętrzna
      - » public
      - » package access
    - klasa wewnętrzna dowolny
  - Wiedza o klasie otaczającej
    - Możliwość kontaktu z nią

Klasy wewnętrzne cd.

- Przykład2_3
  - Można wykonać bez klas wewnętrznych
- Przykład2_4
  - Typowe zastosowanie klas wewnętrznych
    - » Metody zwracają referencje obiektów klas wewnętrznych
    - » Wielokrotne dziedziczenie
    - » Dostęp do klasy wewnętrznej poprzez NazwaKlasyZewnetrznej.NazwaKlasyWewnętrznej
- Przykład2_5
  - Dostęp do pól/metod klasy zewnętrznej

Klasy wewnętrzne cd.

- Wewnętrzne klasy lokalne
  - Ograniczenie widoczności klasy

```
public class A
{ void metoda(){ …
    if (…)
    {
      class lokalna{ …}
    } …
}}
```

  - klasa lokalna

Anonimowe klasy wewnętrzne

- Czy zawsze potrzebujemy nazw klas ?
  - Nie jeżeli:
    - Klasa tylko tymczasowa niedostępna zewnątrz
    - Klasa lokalna potrzebna wewnątrz metody
    - Następuje rzutowanie na inną klasę
  - Przyklad2_6
    - Klasa anonimowa może zawierać
      - konstruktor
      - przeładowanie pól/metod

Anonimowe klasy wew. cd.

- Klasa anonimowa ograniczenia
  - Może korzystać Tylko ze zmiennych final
  - Tylko jeden konstruktor
  - Mogą implementować TYLKO jeden interfejs
- Przykład2_7

Dziedziczenie klas wewnętrznych

- Przykład2_8
  - Klasa wewnętrzna – oddzielny byt
    - Inna przestrzeń nazw
- Przykład2_9
  - Dziedziczenie
    - Tylko jawne wywołanie przy dziedziczeniu

Własności klas wewnętrznych

- Referencja do obiektu klasy nadrzędnej
  - NazwaKlasyZewnętrznej.this
  - Inicjalizacja obiektu klasy wewnętrznej
    - Musi istnieć obiekt klasy zewnętrznej
  - Dostęp do pól/metod klasy zewnętrznej
  - Nie może posiadać metod/pól statycznych
- Można bez referencji
  - Statyczna klasa wewnętrzna
    - zwana klasą zagnieżdżoną (*nessted class*)

Klasy zagnieżdżone

- Brak referencji do obiektu klasy zewnętrznej
  - Nie potrzeba obiektu klasy zewnętrznej
  - Brak dostępu do elementów klasy zewnętrznej
  - Może posiadać pola/metody statyczne

Błędy w programach

- Rodzaje błędów
  - Wykrywane na etapie kompilacji
    - Brak zaimportowanych bibliotek
    - Błędy składni
  - Wykrywane w trakcie wykonywania
    - Przekroczenie rozmiaru tablicy
    - Brak prawa do zapisu pliku
    - Utrata łączności (sieć)

Wyjątki zarys zastosowania

- W trakcie wykonania programu
  - Jeżeli nastąpi błąd
    - Program zwraca obiekt (wyjątek)
      - Decyzja co zrobić
        - » Poprawić działanie, uniknąć konfliktu
        - » Utworzyć kopię danych
        - » Bezpiecznie zakończyć program
- Wyjątki jedyny oficjalny sposób na takie błędy w Javie

Tradycyjna obsługa błędów

- Tradycyjne programowanie
  - Zgłaszanie błędów poprzez flagi
    - Obowiązek sprawdzania flag
      - Nagminne „zapominanie” o błędach
    - Nieczytelność kodu
      - Wymieszanie programu i obsługi błędów

Obsługa błędów w Javie

- Wyjątki w Javie
  - Wyrzucanie wyjątków przy błędach
    - Przerwanie „na chwilkę” wykonywania programu
    - Obsługa sytuacji wyjątkowej
  - Wymuszenie obsługi błedów
  - Podział kodu
    - Część normalna programu
    - Obsługa błędów

Standardowe Wyjątki

- Klasa Thrownable
- Dziedziczą po Thrownable
  - Error
    - Dla nas błędy pomijalne
      - Błędy kompilacji
      - Błędy systemu
  - Exception
    - Dla nas błędy znaczące

Standardowe Wyjątki cd.

- Exception
  - Wyjątki
    - java.io.IOException
    - Sprawdzane podczas wykonywania
  - Wyjątki niesprawdzalne (*unchecked*)
    - RuntimeException
      - NullPonterException, ArrayIndexOutOfBoundsException
    - Nie sprawdzane podczas wykonywania
    - Przeważnie błąd programisty

Wyjątki obsługa

- „Podejrzane” kawałki kodu
  - blok
    - try{ … }
  - Co jeżeli wyrzuci wyjątek
    - catch (NazwaWyjatku e) { … }
      - Dla każdego typu wyjątku z osobna
    - finally { … }
      - Przywrócenie systemu do stanu przed try
        - » Zamknięcie plików itp..
      - Wykonywane zawsze
- Przykład 2_10

Definiowanie wyjątków

- Własne wyjątki
  - Dziedziczą po Exception
    - class MojWyjatek extends Exception {}
    - Najczęściej ważna jest tylko nazwa
    - Można przeładować metody klasy Exception
- Informowanie o możliwych wyjątkach
    - Metody określają listę możliwych wyjątków

```
public void metoda() throws Wyjatek1, Wyjatek2 {…}
```

Przykład 2_11

Informacje zawarte w wyjątkach

- Kolejność wyłapywania wyjątków
  - Pamiętajmy o polimorfizmie
  - Przykład 2_12
- Standardowe metody wyjątków
  - konstruktory
    - Exception (), Exception (String)
  - Osiągalne informacje
    - getMessage(), toString(), printStackTrace()
  - Przykład 2_13

Rethrowing – przekazywanie wyjątków

- Nie wiemy co zrobić z wyjątkiem
  - Ignorujemy ??
    - » Naganne zachowanie
  - Przekazujemy dalej (*rethrowing*)
    - » Może później będzie nam łatwiej ☺
    - – Jako ten sam wyjątek
    - – Jako inny lub RuntimeException
  - Przykład 2_14
  - Przykład 2_15
  - Przykład 2_16

Przypadki szczególne

- Ostrożnie z konstruktorami
  - Pamięć wyczyści JVM
  - Co z otwartymi plikami, połączeniami sieciowymi??
- finally jest wykonywane ZAWSZE
  - Przykład:
    - Otwieramy plik i próbujemy zapisać:
      - » Możliwe wyjątki: FileNotFoundException i inne:
    - Plik należy zamknąć
      - » Ale przy FileNotFoundException niema co zamykać
    - Nie zamykamy pliku w finally

Rozwiązywanie problemów

1. Wyłapuj wyjątki w odpowiednich miejscach
   - Staraj się go nie przechwytywać jak nie wiesz co zrobić
2. Popraw „błąd” i wróć z powrotem
3. Załataj co się da i idź dalej
4. Spróbuj zrobić to inaczej
5. Przekaż wyjątek dalej
6. Zakończ program 😐
7. Przemyśl logikę programu