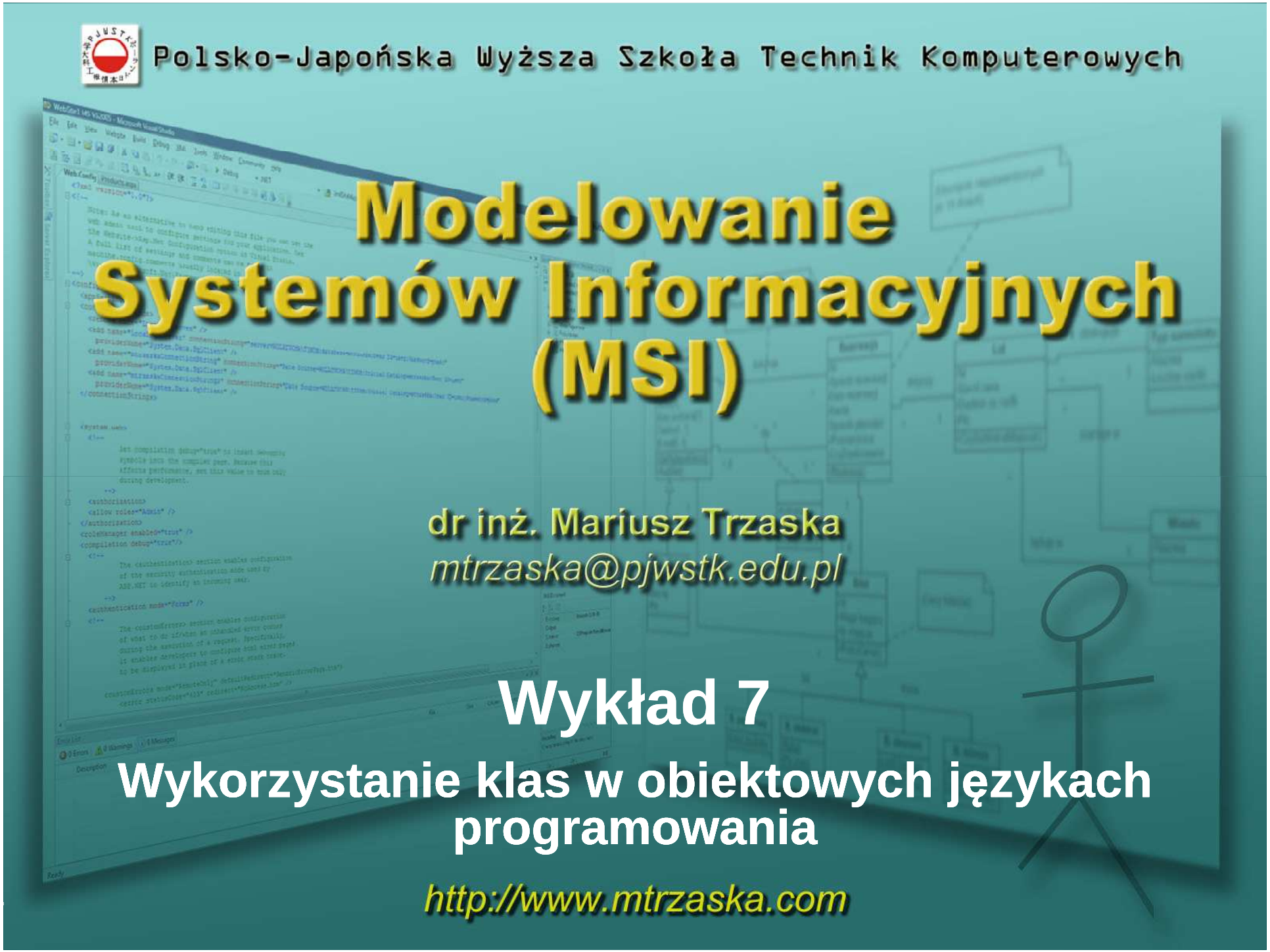

Polsko-Japońska Wyższa Szkoła Technik Komputerowych
Modelowanie Systemów Informacyjnych (MSI)
dr inż. Mariusz Trzaska
mtrzaska@pjwstk.edu.pl
Wykład 7
Wykorzystanie klas w obiektowych językach programowania
http://www.mtrzaska.com

Zagadnienia

- Realizacja w popularnych językach programowania (Java) poniższych konstrukcji:
  - Klasa
  - Ekstensja
  - Atrybuty:
    - proste i złożone
    - wymagane i opcjonalne
    - pojedyncze i powtarzalne
    - obiektu i klasowe
    - Wyliczalne (pochodny)
  - Metody:
    - obiektu
    - klasowe
    - przesłonięcie i przeciążenie
  - Trwałość ekstensji
  - Klasa dedykowana
- Podsumowanie

Klasa

- Klasa jest nazwanym opisem grupy obiektów, które współdzielą ten sam zbiór własności (inwariantów).
- Klasa **nie** jest zbiorem obiektów,
- Stosunek klasa/podklasa oznacza, że obiekty podklasy posiadają wszystkie inwarianty nadklasy, plus (ewentualnie) inwarianty swoje. Np. klasa Student ma wszystkie inwarianty klasy Osoba, plus inwarianty własne.
- **Klasa opisuje obiekt.**

Wykłady z PRI autorstwa dr inż. Ewy Stemposz oraz prof. Kazimierza Subiety.

Atrybuty

- Służą do opisu własności obiektów
- Rodzaje
  - Proste i złożone
  - Wymagane i opcjonalne
  - Pojedyncze i powtarzalne
  - Obiektu i klasowe
  - Wyliczalne (pochodne)

Metody

- Umożliwiają wykonywanie operacji na obiektach, prowadzących (zwykle) do zmiany ich stanu.
- Rodzaje
  - **Metoda obiektu**. Wykonuje operacje (oraz ma dostęp) na jednym, konkretnym obiekcie. Na tym, na rzecz, którego została wywołana.
  - **Metoda klasowa**. Ma dostęp do całej ekstensji klasy. Wywoływana jest na rzecz klasy, a nie obiektu. Dzięki temu można jej użyć nawet nie mając żadnego obiektu danej klasy.

Klasy, a języki programowania

- Jak ma się podana definicja do popularnych, obiektowych języków programowania?
- W językach
  - Java,
  - MS C#,
  - C++

  klasy występują w sposób zgodny z przytoczoną definicją.
- Niestety nie dotyczy to wszystkich pojęć znanych z obiektowości (UML).

Klasy w języku Java

- Załóżmy, że potrzebna nam jest klasa opisująca film w wypożyczalni wideo.

```
**
 * Informacje o filmie.
 *
*/
public class Film {
        /* Cialo klasy */
}
```

Ekstensja klasy

- Zbiór aktualnie istniejących obiektów danej klasy.
- W językach
  - Java,
  - MS C#,
  - C++

  ekstensja klasy nie występuje.
- Co w takim razie możemy zrobić?
- Własna implementacja ekstensji klasy

Implementacja ekstensji klasy

- Dwa różne podejścia. Implementacja:
  - W ramach tej samej klasy biznesowej
  - Przy użyciu klasy dodatkowej
    - Klasa **Film**, jej ekstensja np. **Filmy**
    - Klasa **Film**, jej ekstensja np. **FilmEkstensja**
- Które podejście jest lepsze?
  - Wady
  - Zalety

Implementacja ekstensji klasy (2)

- Implementacja w ramach tej samej klasy
  - Kontener przechowujący referencje do obiektów danej klasy (jako atrybut klasowy czyli `static`),
  - Metody pomocnicze
    - Dodanie,
    - Usunięcie,
    - Wyszukanie.
    - ?
  - Realizacja metod klasowych. W tej samej klasie, ze słowem kluczowym `static`.

Implementacja w ramach tej samej klasy

```
public class Film {
     // Implementacja czesci biznesowej
     public Film() {
          // Dodaj do ekstensji
          dodajFilm(this);
     }

     // Implementacja ekstensji

     /** Ekstensja. */
     private static Vector<Film> ekstensja = new Vector<Film>();

     private static void dodajFilm(Film film) {
          ekstensja.add(film);
     }
     private static void usunFilm(Film film) {
          ekstensja.remove(film);
     }
     // …
}
```

Implementacja w ramach tej samej klasy (2)

```
public class Film {
    // c. d.
    /** Wyswietla ekstensje. Metoda klasowa */
    public static void pokazEkstensje() {
        System.out.println("Ekstensja klasy Film: ");
        for(Film film : ekstensja) {
                System.out.println(film);
        }
    }
}
```

```
public static void main(String[] args) {
    // Test: Implementacja ekstensji w ramach tej samej klasy
    Film film1 = new Film();
    Film film2 = new Film();

    Film.pokazEkstensje();
}
```

Ekstensja klasy Film:

mt.mas.Film@126804e

mt.mas.Film@b1b4c3

Implementacja ekstensji klasy (3)

- Implementacja przy użyciu klasy dodatkowej
  - Nazewnictwo
    - Klasa Film, jej ekstensja np. Filmy
    - Klasa Film, jej ekstensja np. FilmEkstensja
  - Możliwość stworzenia wielu różnych ekstensji
  - Kolekcja przechowująca referencje do obiektów klasy biznesowej,
  - Metody pomocnicze
    - Dodanie,
    - Usunięcie,
    - Wyszukanie.

Implementacja ekstensji klasy (4)

- Implementacja przy użyciu klasy dodatkowej – c. d.
  - Realizacja metod klasowych. Będą na zewnątrz klasy biznesowej – w klasie zarządzającej ekstensją.
  - Problemy z dostępem do atrybutów (public/private/protected).
  - Utrudnione automatyczne dodawanie do ekstensji.
  - Inne rozwiązania, np.
    - z klasą wewnętrzną
    - Kolekcją referencji jako atrybutem `static`

Implementacja ekstensji przy użyciu klasy dodatkowej

```
public class Film {
        /* Cialo klasy */
}
```

```
public class FilmEkstensja {
     private Vector<Film> ekstensja = new Vector<Film>();

     public void dodajFilm(Film film) {
          ekstensja.add(film);
     }
     public void usunFilm(Film film) {
          ekstensja.remove(film);
     }
     public void pokazEkstensje() {
          System.out.println("Ekstensja klasy Film: ");
          for(Film film : ekstensja) {
                    System.out.println(film);
          }
     }
}
```

Implementacja ekstensji przy użyciu klasy dodatkowej (2)

```
public static void main(String[] args) {
    // Test: Implementacja ekstensji przy uzyciu klasy dodatkowej

    FilmEkstensja filmEkstensja = new FilmEkstensja();

    Film film1 = new Film();
    filmEkstensja.dodajFilm(film1);

    Film film2 = new Film();
    filmEkstensja.dodajFilm(film2);

    filmEkstensja.pokazEkstensje();
}
```

Ekstensja klasy Film:
mt.mas.Film@d2906a
mt.mas.Film@72ffb

Atrybuty w obiektowości, a w języku Java

- Rodzaje
  - **Proste**. Występują w takiej postaci jak w obiektowości (w UML).

```
public class Film {
     private float cena;
}
```

  - **Złożone**. Atrybut złożony jest opisywany za pomocą nowej klasy (np. data). Konsekwencje:
    - W klasie biznesowej (np. Film) przechowujemy referencję do jego wystąpienia, a nie (złożoną) wartość.
    - W związku z powyższym możemy go współdzielić (inaczej niż w „teoretycznej” semantyce atrybutu złożonego), np. inny obiekt może przechowywać referencję do tej samej daty.

```
public class Film {
     private Date dataDodania;
}
```

Atrybuty w obiektowości, a w języku Java (2)

- Rodzaje – c. d.
  - **Wymagane**.
    - Każdy **atrybut prosty** przechowuje jakąś wartość – nie może nie przechowywać.
    - **Atrybut złożony** przechowuje referencję do obiektu „będącego jego wartością”. Ponieważ jest to referencja, może mieć wartość `null`, czyli „brak wartości”. Należy „ręcznie” sprawdzać czy jest różna od `null`.
  - **Opcjonalne**
    - Dla **atrybutów złożonych** przechowujemy null jako informację o braku wartości.
    - Co z **atrybutami prostymi**? Klasy opakowujące!

Atrybuty w obiektowości, a w języku Java (3)

- Rodzaje – c. d.
  - **Pojedyncze**. Taka sama semantyka jak w obiektowości (w UML).
  - **Powtarzalne**. Należy wykorzystać tablice lub kontenery (rozwiązanie preferowane gdy liczba wartości jest zmienna).
  - **Obiektu**. Taka sama semantyka jak w obiektowości (w UML).
  - **Klasowe**. Sposób realizacji zależy od podejścia do ekstensji:
    - Ekstensja w ramach tej samej klasy ➔ atrybuty klasowe w tej samej klasie ze słowem kluczowym `static`,
    - Ekstensja jako klasa dodatkowa ➔ atrybuty klasowe w klasie dodatkowej (bez słowa kluczowego `static`).

Atrybuty w obiektowości, a w języku Java (3)

- Rodzaje – c. d.
  - **Wyliczalne** (pochodne).
    - W przypadku:
      - hermetyzacji ortodoksyjnej, specjalne traktowanie atrybutu zaimplementowane w metodzie udostępniającej/zmieniającej jego wartość.
      - bezpośredniego dostępu do atrybutu, implementacja jego specjalnego traktowania jest mocno utrudniona (niemożliwa?).
    - Doskonały mechanizm: *properties* jak na razie tylko w języku MS C#.

```
private float cena {
    get {
        return cena_netto * 1.22;
    }
    set {
        cena_netto = value / 1.22;
    }
}
```

Metody w obiektowości, a w języku Java

- Rodzaje
  - **Metoda obiektu**. Taka sama semantyka jak w obiektowości (w UML).

```
public float getCena() {
    return cena;
}
```

  - **Metoda klasowa**. Sposób realizacji zależy od podejścia do ekstensji (np. `void pokazEkstensje()`):
    - Ekstensja w ramach tej samej klasy ➔ metody klasowe w tej samej klasie ze słowem kluczowym `static`,
    - Ekstensja jako klasa dodatkowa ➔ metody klasowe w klasie dodatkowej (bez słowa kluczowego `static`).

Metody w obiektowości, a w języku Java (2)

- **Przeciążenie metody**. Taka sama semantyka jak w obiektowości (w UML).

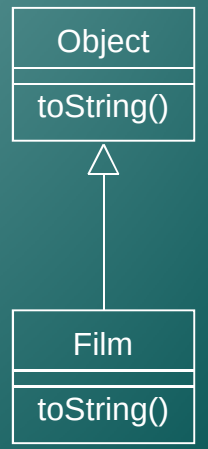
```
public float getCena() {
     return cena;
}
public float getCena(float stawkaVAT) {
     return cena * (1.0f + stawkaVAT / 100.0f);
}
```

- **Przesłonięcie metody**. Taka sama semantyka jak w obiektowości (w UML).

```
public class Film {
     // Implementacja czesci biznesowej

     public String toString() {
          return "Film: " + tytul;
     }
}
```

Ekstensja klasy Film:
mt.mas.Film@126804e
4c3

Ekstensja klasy Film:
Film: Terminator 1
Film: Terminator 2



Trwałość ekstensji

- Ekstensja klasy jest trwała gdy jej obiekty „przeżyją” wyłączenie systemu – po ponownym włączeniu systemu będziemy mieli te same obiekty.
- W językach
  - Java,
  - MS C#,
  - C++

  cecha ta nie występuje bezpośrednio.
- W związku z tym implementujemy ja ręcznie zapamiętując ekstensje na dysku, a następnie je wczytując.

Trwałość ekstensji (2)

- W efekcie, zamiast **tych samych obiektów**, mamy **takie same obiekty**.
- Implementacja
  - Ręczna,
    - Szybkość,
    - Duża kontrola nad efektem końcowym,
    - Większa odporność na zmiany w kodzie utrwalanych klas,
    - Mały plik,
    - Wymaga (przeważnie) sporo pracy.
  - Korzystająca z serializacji,
    - Bardzo łatwa w użyciu,
    - Mniejsza szybkość,
    - Duży plik,
    - Możliwość kontroli dzięki:
      - dodaniu metod:
        - `private void writeObject(ObjectOutputStream stream) throws IOException`
        - `private void readObject(ObjectInputStream stream) throws IOException, ClassNotFoundException`
      - słowu kluczowemu `transient`.

Trwałość ekstensji (3)

- Implementacja – c. d.
  - W oparciu o bazę danych,
    - Konieczność mapowania struktur języka Java na konstrukcje z bazy danych,
    - Możliwość skorzystania z języka zapytań,
    - Praca z różnymi bazami danych,
    - Duży plik (czasami),
    - Szybkość działania
  - Korzystając z gotowych bibliotek.
    - Hibernate (*http://www.hibernate.org/*)
    - Java Persistence API (*https://glassfish.dev.java.net/*)
    - Java Data Objects (*http://www.jpox.org/*)
    - …

Trwałość ekstensji – implementacja ręczna

- W każdej z klas biznesowych znajduje się metoda zapisująca oraz odczytująca stan pojedynczego obiektu.

```
public class Film {
     private String tytul;
     private float cena;
     private Date dataDodania;

     private void write(DataOutputStream stream) throws IOException {
          stream.writeUTF(tytul);
          stream.writeFloat(cena);
          stream.writeLong(dataDodania.getTime());
     }

     private void read(DataInputStream stream) throws IOException {
          tytul = stream.readUTF();
          cena = stream.readFloat();
          long time = stream.readLong();
          dataDodania = new Date(time);
     }
}
```

Trwałość ekstensji – implementacja ręczna (2)

- W każdej z klas zarządzających ekstensją (lub w klasie biznesowej jeżeli wybraliśmy takie podejście), znajdują się metody zapisujące oraz odczytujące ekstensję.

```
public class Film {
     public static void zapiszEkstensje(DataOutputStream stream) throws IOException {
          stream.writeInt(ekstensja.size());
          for(Film film : ekstensja) {
               film.write(stream);
          }
     }

     public static void odczytajEkstensje(DataInputStream stream) throws IOException {
          Film film = null;
          int liczbaObiektow = stream.readInt();
          ekstensja.clear();
          for(int i = 0; i < liczbaObiektow; i++) {
               film = new Film();
               film.read(stream);
          }
     }
}
```

Trwałość ekstensji – implementacja ręczna (3)

- Przykład pokazuje zapis oraz odczyt ekstensji.
- Dzięki podzieleniu funkcjonalności na odpowiednie metody, możemy w łatwy sposób zapisywać inne ekstensje do tego samego strumienia (pliku).
- Dzięki temu, stan całej aplikacji może być zapamiętany w jednym pliku.
- Efekt działania
  - Ekstensja utworzona w pamięci,
  - Ekstensja zapisana i odczytana z pliku

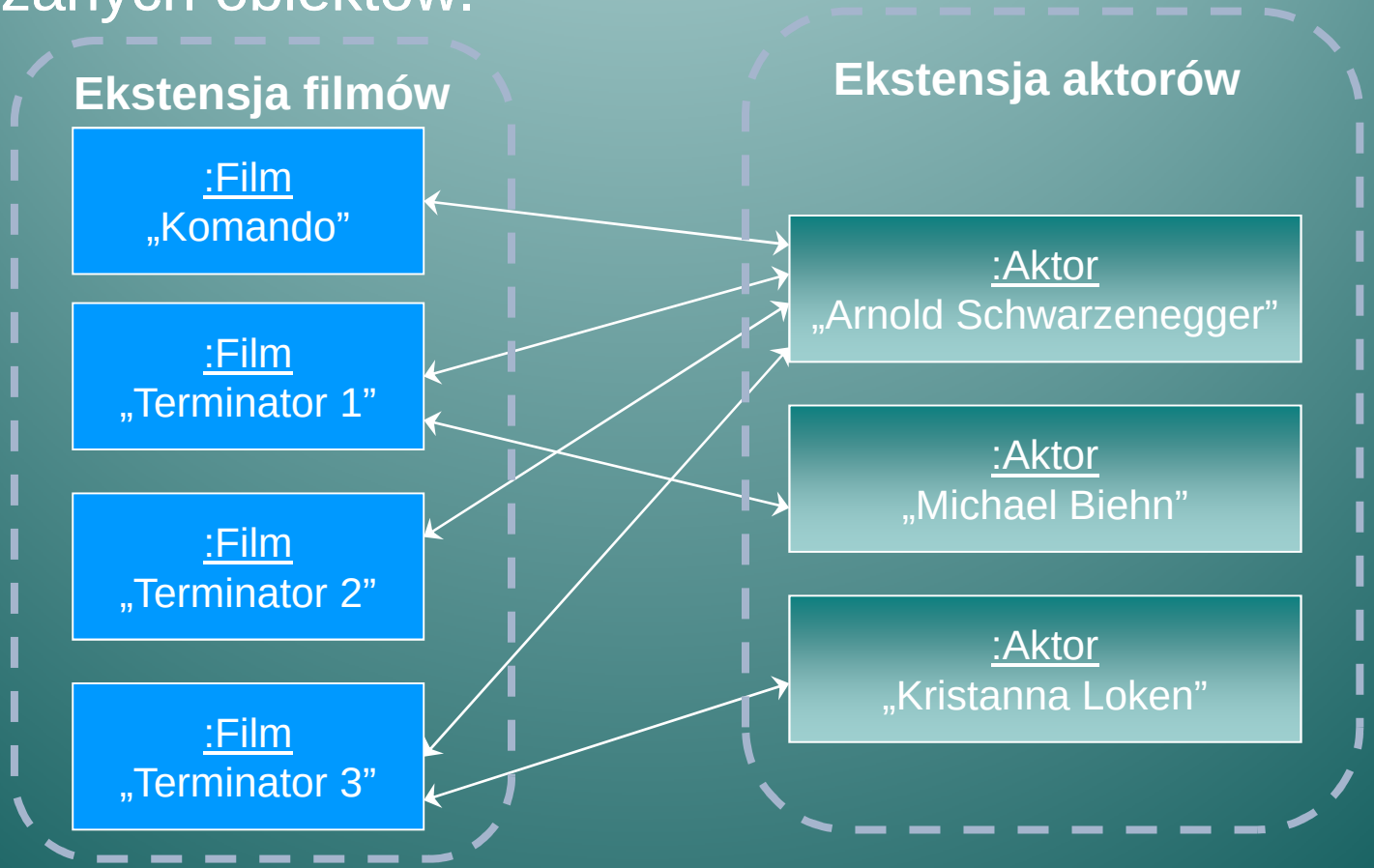
```
Ekstensja klasy Film:
Film: Terminator 1, id: mt.mas.Film@27391d
Film: Terminator 2, id: mt.mas.Film@116ab4e
Ekstensja klasy Film:
Film: Terminator 1, id: mt.mas.Film@1434234
Film: Terminator 2, id: mt.mas.Film@af8358
```

Trwałość ekstensji – implementacja ręczna (4)

```
final String ekstensjaPlik = "d:\\Ekstensja.bin";

Film film1 = new Film("Terminator 1", new Date(), 29.90f);
Film film2 = new Film("Terminator 2", new Date(), 34.90f);

try {
    // Zapisz ekstensje do strumienia
    DataOutputStream out2 = new DataOutputStream(new
        BufferedOutputStream(new FileOutputStream(ekstensjaPlik)));
    Film.zapiszEkstensje(out2);
    out2.close();

    // Odczytaj ekstensje ze strumienia
    DataInputStream in2 = new DataInputStream(new BufferedInputStream(new
        FileInputStream(ekstensjaPlik)));
    Film.odczytajEkstensje(in2);
    in2.close();
} catch ( … ) {
    // …
}
```

Trwałość ekstensji – problem z powiązaniami

- Przedstawiony sposób ręcznej implementacji trwałości jest bardzo prosty. Nie pozwala na właściwe traktowanie powiązanych obiektów.



Trwałość ekstensji – implementacja z wykorzystaniem serializacji

- Serializacja jest mechanizmem zaimplementowanym w ramach bibliotek języka Java.
- Umożliwia automatyczne:
  - zapisywanie grafu obiektów do strumienia,
  - Odczytywanie grafu obiektów ze strumienia.
- Jedynym wymogiem, który trzeba spełnić aby z niego korzystać, jest „specjalna” implementacja przez klasę (oraz wszystkie jej elementy składowe) interfejsu `Serializable`.
- Specjalność implementacji interfejsu polega na tym, że w najprostszym przypadku deklarujemy jego implementację przez klasę, ale nie musimy implementować jego metod. Tym zajmie się „kompilator” języka Java.

Trwałość ekstensji – implementacja z wykorzystaniem serializacji (2)

- Deklarujemy implementację interfejsu przez klasę biznesową

```
public class Film implements Serializable {
     private String tytul;
     private float cena;
     private Date dataDodania;
}
```

- Tworzymy metody do zapisu oraz odczytu ekstensji

```
public class Film implements Serializable {

     public static void zapiszEkstensje(ObjectOutputStream stream) throws IOException {
          stream.writeObject(ekstensja);
     }

     public static void odczytajEkstensje(ObjectInputStream stream) throws IOException {
          ekstensja = (Vector<Film>) stream.readObject();
     }
}
```

Trwałość ekstensji – implementacja z wykorzystaniem serializacji (3)

- Użycie

```
try {
    // Zapisz ekstensje do strumienia
    ObjectOutputStream out = new ObjectOutputStream(new
    FileOutputStream(ekstensjaPlik));
    Film.zapiszEkstensje(out);

    // Odczytaj ekstensje ze strumienia
    ObjectInputStream in = new ObjectInputStream(new FileInputStream(ekstensjaPlik));
    Film.odczytajEkstensje(in);
} catch (FileNotFoundException e) { ...
```

- Rozmiar pliku z ekstensją:
  - Ręczna implementacja: 56 bajtów,
  - Serializacja: 354 bajty.
  - Dla większych danych różnice są mniejsze – około x2.

```
Ekstensja klasy Film:
Film: Terminator 1, id: mt.mas.Film@148aa23
Film: Terminator 2, id: mt.mas.Film@199f91c
Ekstensja klasy Film:
Film: Terminator 1, id: mt.mas.Film@92bbba
Film: Terminator 2, id: mt.mas.Film@162dbb6
```

Zarządzanie ekstensją

- Przedstawione sposoby implementacji zarządzania ekstensją będą (prawie) takie same dla każdej biznesowej klasy w systemie.
- Czy da się to jakoś zunifikować? Aby nie pisać wiele razy (prawie) tego samego kodu?
- Oczywiście – wykorzystamy dziedziczenie istniejące w języku Java.

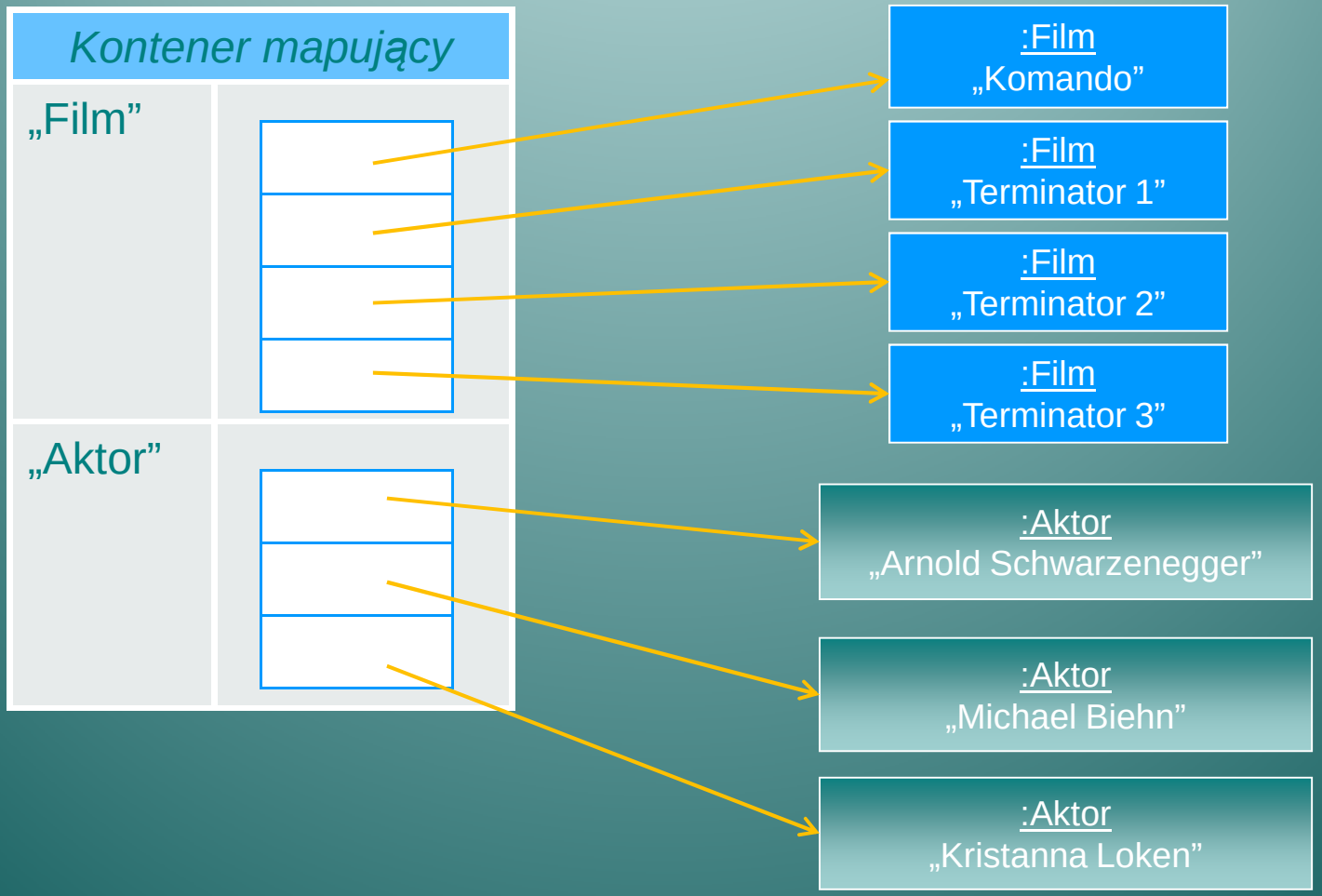
Uniwersalne zarządzanie ekstensją

- Stworzymy klasę z której będą dziedziczyć wszystkie biznesowe klasy w naszej aplikacji.
- Nazwijmy ją `ObjectPlus` i wyposażmy w:
  - trwałość,
  - zarządzanie ekstensją,
  - ?
- Zastosujemy pierwsze z omawianych podejść do implementacji ekstensji: w ramach tej samej klasy.

Uniwersalna ekstensja

- Ponieważ wszystkie biznesowe klasy dziedziczą z jednej nadklasy (ObjectPlus), nie możemy zastosować zwykłego kontenera przechowującego referencje.
- Użyjemy kontenera przechowującego klucze i wartości:
  - Kluczem będzie nazwa konkretnej biznesowej klasy,
  - Wartością kontener zawierający referencje do jej wystąpień (właściwa ekstensja).
- Innymi słowy, ten nowy kontener będzie zawierał wiele ekstensji, a nie jedną ekstensję.

Uniwersalna ekstensja (2)



Uniwersalna ekstensja (3)

- Konstruktor każdej z klasy biznesowych będzie odwoływał się do konstruktora z nadklasy.

```
public class Film2 extends ObjectPlus implements Serializable {
     private String tytul;
     private float cena;
     private Date dataDodania;

     /**
      * Konstruktor.
      */
     public Film2(String tytul, Date dataDodania, float cena) {
          // Wywołaj konstruktor z nadklasy
          super();

          this.tytul = tytul;
          this.dataDodania= dataDodania;
          this.cena = cena;
     }

     // Dalsza implementacja czesci biznesowej
}
```

Uniwersalna ekstensja (4)

- Konstruktor z nadklasy zadba o właściwe dodanie do ekstensji.

```
public class ObjectPlus implements Serializable {
     private static Hashtable ekstensje = new Hashtable();

     public ObjectPlus() {
          Vector ekstensja = null;
          Class klasa = this.getClass();

          if(ekstensje.containsKey(klasa)) {
               // Ekstensja tej klasy istnieje w kolekcji ekstensji
               ekstensja = (Vector) ekstensje.get(klasa);
          }
          else {
               // Ekstensji tej klasy jeszcze nie ma -> dodaj ją
               ekstensja = new Vector();
               ekstensje.put(klasa, ekstensja);
          }

          ekstensja.add(this);
     }
}
```

Uniwersalna ekstensja (5)

- Utrwalenie takich ekstensji też jest bardzo proste (korzystamy z serializacji).

```
public class ObjectPlus implements Serializable {
     private static Hashtable ekstensje = new Hashtable();

     // ...

     public static void zapiszEkstensje(ObjectOutputStream stream) throws IOException {
          stream.writeObject(ekstensje);
     }

     public static void odczytajEkstensje(ObjectInputStream stream) throws IOException {
          ekstensje = (Hashtable) stream.readObject();
     }

     // ...
}
```

Uniwersalne metody klasowe

- Część metod klasowych też może korzystać z ogólnej funkcjonalności zgromadzonej w nadklasie, np. wyświetlenie ekstensji.

```
public class ObjectPlus implements Serializable {
     // ...
     public static void pokazEkstensje(Class klasa) throws Exception {
          Vector ekstensja = null;
          if(ekstensje.containsKey(klasa)) {
               // Ekstensja tej klasy istnieje w kolekcji ekstensji
               ekstensja = (Vector) ekstensje.get(klasa);
          }
          else {
               throw new Exception(„Nieznana klasa " + klasa);
          }
          System.out.println("Ekstensja klasy: " + klasa.getSimpleName());
          for(Object obiekt : ekstensja) {
               System.out.println(obiekt);
          }
     }
     // ...
}
```

Uniwersalne metody klasowe (2)

- Wyświetlenie ekstensji w klasie biznesowej (korzysta z funkcjonalności zdefiniowanej w nadklasie).

```
public class Film2 extends ObjectPlus implements Serializable {
    // ...
    public static void pokazEkstensje() throws Exception {
        ObjectPlus.pokazEkstensje(Film2.class);
    }
}
```

```
Ekstensja klasy: Film2
Film2: Terminator 1, id: mt.mas.Film2@199f91c
Film2: Terminator 2, id: mt.mas.Film2@1b1aa65
Ekstensja klasy: Film2
Film2: Terminator 1, id: mt.mas.Film2@4ac00c
Film2: Terminator 2, id: mt.mas.Film2@1865b28
```

Podsumowanie

- Część pojęć z obiektowości występuje w popularnych językach programowania.
- Niestety, niektóre z nich istnieją w niepełnym zakresie lub nie ma ich w ogóle.
- W większości przypadków, nieistniejące konstrukcje można:
  - zaimplementować samodzielnie na kilka, różnych sposobów,
  - Obsłużyć korzystając z gotowych bibliotek.
- Całą funkcjonalność związaną z zarządzaniem ekstensją klasy, warto zgromadzić w specjalnej nadklasie.