MP01

Ekstensja: zbiór aktualnie występujących obiektów danej klasy.

W ramach tej samej klasy: wektor (kontener), który przechowuje referencje do obiektów danej klasy, metody klasowe (static) np. dodaj i usuń film.

W ramach inne klasy, podobnie tylko ze jest możliwość stworzenia kilku ekstensji.

Trwałość ekstensji: Ekstensja jest trwała gdy obiekty jej klasy 'przeżyją' wyłączenie systemu (np. dzięki serializacji).

Atrybut złożony: opisywany za pomocą nowej klasy: atrybut_zlozony = **new** Klasa();

Atrybut opcjonalny: dla złożonych atrybutów przypisany jako null, dla prostych klasy opakowujące.

Atrybut powtarzalny: [1..*] atrybut typu listowego (np. ArrayList).

ArrayList<Typ> powtarzalny = new **ArrayList**<Typ>();

Atrybut klasowy: atrybut statyczny, dostępny dla całej klasy.

static String atrybut_klasowy;

Atrybut unikalny: dowolny atrybut z niepowtarzalną wartością. Należy sprawdzać czy taka wartość już nie istnieje.

Atrybut pochodny: atrybut, który jest wyliczany za pomocą metody (np. wiek po dacie urodzenia).

Metoda klasowa: metoda, która jest wykonywana dla całej klasy, nie dla konkretnego obiektu

static wypisz(){};

Przeciążenie metody: polega na stworzeniu kilku metod o takiej samej nazwie z różnymi parametrami wywołania. W zależności od wywołanych parametrów, odpowiednia metoda zostanie wykonana.

Przesłonięcie metody: zapobieganie zwracania adresu do obiektu poprzez metodę. Implementuje się to przy pomocy dodatkowej metody toString(), która zostaje wykonana w przypadku próby zwrócenia adresu do obiektu.

```
class Osoba(){
      ...
      toString(){
         return osoba.imie + ” ” + osoba nazwisko;
      }
}
```

MP02

Asocjacja binarna: asocjacja 1 do 1. Przykład: mamy 2 klasy, Pizza i Pudelko, w klasie Pudelko tworzymy atrybut Pizza (public Pizza pizza) i w konstruktorze umieszczamy go jako parametr:
public Pudelko(int wielkosc, Pizza pizza), w klasie Pizza robimy adekwatnie.

Asocjacja z atrybutem: tworzona poprzez klasę asocjacji, np. PracownikWFirmie.

Asocjacja kwalifikowana: metoda, która na podstawie Klucza (np. nazwy firmy) zwróci nam wartość (np. pracowników, którzy są w niej zatrudnieni). Stosujemy do tego Mapy (**TreeMap**, **HashMap**). Na przykład w metoda znajdzPracownika(Klucz k) wyświetli wszystkich pracowników na podstawie danego klucza (nazwy firmy)

Asocjacja n-arna: asocjacja wiele do wielu. Polega na tym, że w jednej metodzie może znajdować się wiele obiektów konkretnego typu. Na przykład: Ocena ocena i Przedmiot przed. W obiektach Ocena może być wiele przedmiotów, i w obiekcie Przedmiot może być wiele ocen.

Kompozycja: przekazujemy referencje do obiektu powiązanej klasy w konstruktorze np. jesli mamy klasy Samochod i Silnik, to konstruktor klasy Samochod: **public** Samochod(Silnik jakis_silnik).

MP03

Klasa abstrakcyjna: tworzona poprzez dodanie słowa *abstract*: **public abstract class** Osoba{}

Dziedziczenie: tworzone poprzez dodanie extend i klasy: **class** Student **extends class** Osoba

Polimorfizm metod: polega na różnym wykonaniu metody o takiej samej nazwie w zależności od klasy obiektu. Implementacja polega na różnym zdefiniowaniu konkretnej metody w wielu klasach.

Overlapping: Nie występuje w wielu językach. Uzyskiwany poprzez stworzenie jednej klasy, która będzie zawierała w sobie atrybuty z kilku klas (np. Student i Pracownik w jednej klasie Osoba)

Wielodziedziczenie: Dziedziczenie z wielu klas naraz. Nie występuje w większości języków programowania (na pewno nie ma go w Java™). Radzimy sobie z tym poprzez stworzenie dodatkowej klasy, która jest połączeniem dwóch klas z których chcemy dziedziczyć, np. z Wodny i Ladowy tworzymy WodnoLadowy.

Wieloaspektowe: Dziedziczenie ze względu na atrybut. Implementacja następuje poprzez dodanie różnych atrybutów do klas dziedziczących, na przykład: Wodny i Ladowy, które dziedziczą po klasie Pojazd. Wodny ma atrybut '**int** głębokość' na jakiej może pływać, a Ladowy '**int** ilosc_kol'

Dynamiczne: dziedziczenie, w którym obiekt może zmienić typ (poprzez zastąpienie). W implementacji w trakcie tworzenia nowego obiektu, należy przekopiować dane z poprzedniego poprzez **super.atrybut**.