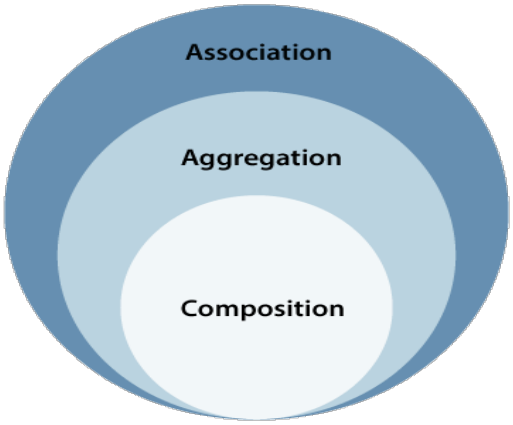
Laborator 3

Rezumat

Agregarea – Obiectul container poate exista si in absenta obiectelor agregate. Este o relatie de tip "*has A*".

Compunerea - Este o agregare puternica, obiectele sunt dependente unele de altele.



Pentru a evidentia diferenta dintre cele doua sa consideram urmatorul exemplu: Avem un editor de text care in implementarea sa are un buffer. Editorul de text lucreaza cu fisiere text(vorbim despre agregare, un fisier text poate exista in absenta unui editor, la fel si cum un editor poate exista in absenta unui fisier text). Cand editorul de text este inchis, bufferul este distrus, dar fisierul text nu. Astfel, existenta bufferului este dependenta de cea a editorului de text(compunere), bufferul nu poate fi alocat daca editorul de text nu este pornit, dar fisierul continua sa existe si dupa inchiderea editorului(agregare).

Mostenirea – Este un procedeu specific programarii orientate pe obiect. O clasa ce mostenste o alta clasa preia atributele si metodele neprivate de la aceasta. Cu alte cuvinte, particularizam un prototip definit deja de o clasa anume, prin particularizare intelegandu-se ca vom crea ceva nou plecand de la ceva existent. Spre exemplu: Avem definite clasele Punct si Cerc care mosteneste Punct. Ce este un Cerc? Un Punct(multime de puncte) ce mai are in plus o raza. Mostenirea este o relatie de "*is A*".

```
class Punct {
        int x;
        int y;
}

class Cerc extends Punct{
        Punct [] p;
        int r;
}
```

Avem diferentele:

- "is A" - Cercul este o multime de Puncte, ce are in plus o raza.
- "has A" - Un Cerc contine un Punct.

Modificatorii de acces in cadrul mostenirii:

- **private** – membrii clasei nu pot fi mosteniti.
- **default** – mostenirea este posibila doar in interiorul pachetului.
- **public si protected** – atributele si metodele pot fi mostenite atat in interiorul pachetului cat si in afara lui.

Upcasting - Convertirea unei referințe la o clasă derivată într-una a unei clase de bază poartă numele de **upcasting**. Upcasting-ul este facut **automat** și **nu** trebuie declarat explicit de către programator.

Downcasting - Este operația **inversă** upcast-ului și este o conversie explicită de tip în care se merge în **jos** pe ierarhia claselor. Acest cast trebuie facut explicit de catre programator.

Supraincarcarea(overload) – reprezinta posibilitatea de a defini mai multe metode cu acelasi nume in cadrul unei clase pentru a oferi functionalitati in plus. Reguli pentru supraincare:

- Numele metodei trebuie sa fie acelasi
- Trebuie schimbata signatura functiei

Suprascrierea(overriding) – Atunci cand o clasa mostenste o alta clasa ea preia toate metodele neprivate. Java ne pune la dispozitie posibilitatea de a redefini comportamentul unei metode. Rescrierea comportamentului unei metode mostenite, dar pastrand semnatura metodei se numeste suprascriere.

Reguli pentru suprascriere:

- Specificatorul de acces trebuie sa fie la fel sau mai putin restrictiv
- Tipul de return trebuie sa fie acelasi sau care mosteneste tipul de return al metodei initiale(cea mostenita)
- Metoda care suprascrie trebuie sa aiba acelasi nume si aceeasi parametri ca metoda suprascrisa
- Metoda care suprascrie arunca acelasi exceptii, mai putine sau mai generale(Discutam in detaliu acest aspect in cadrul labului de exceptii)

Cuvantul cheie super(hiding fields) – In cadrul mostenirii putem avea urmatoare problema: o zona de memorie de acelasi tip si cu acelasi nume atat in clasa mostenita cat si in cea care o mosteneste. Pentru a face diferenta intre cele doua folosim keywordul **super**, impreuna cu **this**.(this face referire la clasa curenta, iar super la clasa parinte).

Un exemplu pentru clarificare:

```
class Animal {
	String color = "white";
}

class Dog extends Animal {
	String color = "black";

	public void printColor() {
		System.out.println(this.color); // Afiseaza black
		System.out.println(super.color); // Afiseaza white
	}
}
```

Asa cum this() poate fi folosit pentru a se face referire la unul dintre constructorii clasei curente, super() poate fi folosit pentru a face referire la unul dintre constructorii clasei superioare. super() nu poate fi apelat decât pe prima linie a constructorului clasei copil.

Polimorfismul – reprezinta posibilitatea unei instante de a lua forma oricaruia dintre prototipurile superioare celei cu care a fost creat.

ArrayList – este un array redimensionabil(de fiecare data cand este plin, el isi dubleaza dimensiunea).

Principalele metode ale acestuia sunt:

- `add()` - adaugă un element la ArrayList.
- `get()` - accesează elementul de pe o anumită poziție din ArrayList.
- `size()` - returnează numărul de elemente din ArrayList.
- `set()` - modifica un element de pe o anumită poziție din ArrayList.
- `remove()` - șterge un element de pe o anumită poziție din ArrayList.
- `clear()` - șterge toate elementele din ArrayList.

Pentru mai multe detalii despre aceasta clasa, puteti accesa documentatia aici. Un alt link util despre folosirea ArrayList se gaseste aici