Programozz Te Is!

Java

Osztályok 5

Absztrakt Osztályok és Interface-ek

Az Öröklésről még egyszer, röviden

Az öröklés során Szülő – Gyermek viszony alakul ki az Osztályok között. A Gyermek Osztály örökli a Szülő Osztály attribútumait és metódusait. Egy Szülő Osztálynak pedig több Gyermek Osztálya is lehet, de minden Gyermeknek csak egy Szülője:

- **Ezáltal elérhető, hogy a Gyermek Osztályokra jellemző, közös attribútumokat és metódusokat elegendő egyszer definiálni a Szülő Osztályban.** Így rengeteg időt és energiát takaríthatunk meg magunknak, hiszen csökkentjük a redundanciát a forráskódjaink között.

- **Az Öröklés egy másik kiemelten fontos tulajdonsága, hogy a Gyermek Osztályok, a Szülő Osztály szerint csoportosíthatóak**, ezáltal még jobban csökkenthető a redundancia. Mit jelent ez?

Nézzünk egy egyszerű példát:

Szeretnék készíteni egy Nyilvántartó rendszert, ami Embereket képes kezelni. Azonban szeretnénk megkülönböztetni Férfiakat és Nőket.

Ebben az esetben mindenképpen létezik a Férfi és a Nő mint Osztály és létezik a Nyilvántartó Osztály, aminek célja a Férfiak és a Nők tárolása:

1. Ha Öröklés nélkül kellene megoldani a feladatot, akkor a Nyilvántartó Osztálynak kellene legyen egy Listája, mely férfiakat tárol és egy második Listája, mely nőket.
2. **Öröklés segítségével azonban elegendő egyetlen egy Lista is**, ami Embereket képes tárolni, ha az Ember Osztály és a Férfi, Nő Osztályok között Szülő – Gyermek viszonyt definiálunk.

```
public class Human {

}

public class Man extends Human {

}

public class Woman extends Human {

}
```

```
public class Main {

	private ArrayList<Human> humans;


	public static void main(String[] args) {
		Man man = new Man();
		Woman woman = new Woman();


		humans.add(man);
		humans.add(woman);
	}
}
```

A feladathoz tartozó implementáció remekül egyszerűsíthető az Öröklés segítségével. **Azonban bevittünk, ezáltal egy hibát is, de mi lehet ez?**

Bekerült a rendszerbe egy plusz Osztály, amivel alapvetően nem akarunk majd dolgozni, csupán a csoportosíthatóság miatt került bele a programunkba. Ettől függetlenül, lehet az Ember Osztályból példányokat létrehozni, így akár inkonzisztens állapotba is tudom vinni a programomat…

```
Human human = new Human();
```

Absztrakt Osztályok

Erre a helyzetre kíván megoldást nyújtani az úgynevezett Absztrakt Osztály nyelvi elem.

Az Absztrakt Osztály egy olyan Osztály, mely nem példányosítható. A működése, a példányosítást leszámítva, a sima Osztályokkal megegyezik. Nézzük meg a fenti példát most már Absztrakt Osztállyal:

```
public abstract class Human {

}

public class Man extends Human {

}

public class Woman extends Human {
```

```
}

public class Main {

	private ArrayList<Human> humans;

	public static void main(String[] args) {

		Human human = new Human();

		Man man = new Man();
		Woman woman = new Woman();


		humans.add(man);
		humans.add(woman);
	}
}
```

Hogyan alakítható át egy egyszerű Osztály, Absztrakt Osztállyá?

- Az ***ABSTRACT*** kulcsszó segítségével

Az Abstract Osztályban, lehetőség nyílik egy metódus meg nem valósítására:

Ugyanis az Absztrakt Osztály lehetőséget kínál arra, hogy definiáljunk egy metódust, de a konkrét megvalósítást már a Gyermek Osztályra bízzuk! Tehát az Absztrakt Osztályban létrehozom a metódust, ezáltal megjelölve, hogy minden Gyermek Osztály rendelkezik ilyen metódussal, de az implementáció a Gyermek Osztály felelőssége. **Az Ilyen metódust nevezzük Absztrakt Metódusnak.**

Hogyan hozzunk létre ***ABSTRACT*** Metódust?

```
public abstract returnType methodName();
```

SZABÁLY:

Ha a Gyermek Osztály egy Absztrakt Osztály leszármazottja, akkor annak minden absztrakt metódusát kötelező megvalósítania!

Rendben, tehát most már megoldottuk, hogy nem tudjuk példányosítani az Ember Osztályt. Azonban a programom tovább bővül:

Kisgyerekeket, Fiúkat és Lányokat is szeretnénk kezelni úgy, hogy a Fiúknak és a Lányoknak vannak csak a kisgyermekekre jellemző közös tulajdonságaik, amik nem érvényesek a felnőttekre.

Ezt a feladatot hogyan tudnánk hatékonyan megoldani?

A legcélszerűbb az lenne, ha létrehoznánk egy Gyermek nevű Absztrakt Osztályt, ott gyűjtve a közös tulajdonságaikat a fiúknak és lányoknak. Azonban ha így tennénk, akkor a Fiú és a Lány Osztályoknak **már két különböző helyről kellene örökölni:**

- Férfi, Gyermek → **Fiú**
- Nő, Gyermek → **Lány**

Azonban sajnos ez továbbra sem lehetséges.

Interface-ek

Erre a problémára nyújt megoldást az Interface nyelvi elem.
Az Interface nem más mint egy speciális Absztrakt Osztály, melynek:

- Minden metódusa ***ABSTRACT*** és ***PUBLIC***
- Nem lehet attribútuma

További előnye az Absztrakt Osztályokhoz képest, hogy míg egy Osztálynak csak egy Szülő Osztálya lehet, addig Interface-ből bármennyit meg tud valósítani, tehát egyszerre akár többet is.

Tehát egy Interface mindent tud, amit egy Abstract Osztály *(redundancia csökkentése, központosíthatóság)* is és egy elem több Interface-t is tud implementálni.

Nézzük meg a fenti problémát tehát most már Interface-ek segítségével:

```
public abstract class Human {

}
```

```
public class Man extends Human {

}

public class Woman extends Human {

}

public interface IKid {

}

public class Boy extends Man implements IKid {

}


public class Girl extends Woman implements IKid {

}



public class Main {

	private ArrayList<Human> humans;


	public static void main(String[] args) {

		Human human = new Human();

		Man man = new Man();
		Woman woman = new Woman();
		Boy boy = new Boy();
		Girl girl = new Girl();

		IKid kid = new IKid();


		humans.add(man);
		humans.add(woman);
		humans.add(boy);
		humans.add(girl);
	}
}
```

Érdekességek az Interface-ekről:

- Minden metódusa ***PUBLIC*** láthatóságú kell legyen
- Minden metódusa ***ABSTRACT***
- Nem lehet attribútuma