**Inteface**

Olyan speciális "osztály", amely csak absztrakt metódusokat (Java 7) és konstansokat tartalmazhat.

Szemantikailag egy viselkedésminta leírása, amelyet egy másik osztály valósít meg.

Nem példányosítható.

Nem leszármaztatással és az absztrakt metódusok felüldefiniálásával használjuk fel, mint az absztrakt osztályt, hanem implementálással.

Implementálás: az interfészben (interface) deklarált metódusok megvalósítása.

Egy osztály tetszőleges számú interfészt implementálhat. Minden interfész implicite absztrakt, nem kell kiírni az abstract kulcsszót.

Az interfész minden metódusa implicite absztrakt, akkor is ha nincs kiírva az abstract kulcsszó, és hozzáférési módosítójuk public. Nincs konstruktora.

Interfészek között lehet többszörös öröklés. nterfész módosítók: public, protected, private, static. A public kivételével a többi csak beágyazott interfész esetén használható.

Egy fordítási egységben csak egy public interfész lehet.

Az interfész neve tetszőleges, az osztálynévhez hasonlóan nagy kezdőbetűs azonosító.

Extends rész: extends szülőinterfészek Több szülő interfész is lehet vesszővel elválasztva (többszörös öröklés).

Szemantikailag a leszármazott egy összetett viselkedésminta, amely a szülő viselkedésmintákból plusz a saját viselkedésmintából tevődik össze. Csak konstansai és abstract tagjai lehetnek (Java 7). – Konstans: automatikusan public static final módosítóval rendelkeznek, de szokás kiírni a módosítókat. – Metódus: automatikusan public abstract módosítóval rendelkeznek. A public kulcsszó kiírható, az abstract nem ajánlott.

– Osztály v. interfész tag: automatikusan public static. Interfészek: interface nevek vesszővel elválasztva.

Ha egy osztály implementál egy interface-t, akkor meg kell valósítani az interface-ben levő összes metódust, az örökölteket is (kivéve, ha az osztály abstract).

Több interfész megvalósításakor lehetnek-e ütközések? Ha a megvalósítandó metódusok azonos szignatúrával rendelkeznek, akkor tulajdonképpen azonosak, tehát csak egy implementáció szükséges. Ha azonos a nevük és paraméterszignatúrájuk, de a visszatérési típusuk eltérő, fordítási hibát kapunk. Default metódus Az interfészben definiálható default implementációval rendelkező public metódus.

Megadása: default módosítóval. Ez a definíció a leszármazott interfészben

– öröklődik default metódusként,

– újradefiniálható default metódusként,

– újradeklarálható abstract metódusként.

Static metódus Az interfészhez (és nem az interfészt megvalósító osztály példányához) kötődő tevékenység. Módosítójuk: public static. A kód jobban áttekinthető, ha ezeket a metódusokat nem külön osztályban definiáljuk. Interfész lehet egy referencia változó statikus típusa.

Interfész típusú referencia értékül minden olyan osztály objektumának a referenciáját felveheti, amely megvalósítja az interfészt. Az interface referenciákra ugyanazok a szabályok érvényesek, mint az objektum referenciákra, kiegészítve azzal, hogy a dinamikus típusa mindig objektum referencia.

Az interface referenciákra is igaz, hogy a referencia statikus típusa határozza meg milyen műveleteket végezhetek a referenciával hivatkozott objektummal. Azaz csak az osztály azon tagjaira hivatkozhatunk, amelyek az interfészben lettek deklarálva !!!

Az instanceof operátor működik interface referenciákra is: ref instanceof InterfaceTípus akkor ad igazat, ha a ref dinamikus típusa implementálja az InterfaceTípus-t. Mire használjuk az interfészt? Protokollt biztosít különböző osztályok között. Pl. Írható olyan fv, amely interfész típusú paramétert vár. Ekkor a fv-nek bármely olyan objektumot átadhatunk, amelynek osztálya implementálja az adott interfészt. Az interfész leírja, hogy egy objektum milyen viselkedési formákkal rendelkezhet. Ezek konkrét megvalósítását az interfészt implementáló osztályok tartalmazzák. Az interfész szerződés az interfész és az implementáló osztályok között, aminek teljesülését a fordító ellenőrzi.

Szerződés alapú tervezés Szerződés: „egy szoftverelem tulajdonságainak olyan specifikációja, amely befolyásolja, hogy potenciális kliensei hogyan használhatják”. A szerződés alapú tervezést (design by contract), mint szoftvertervezési módszert Bertrand Meyer, az Eiffel programozási nyelv megalkotója fejlesztette ki a nyelv részeként az 1980-as közepén. Megközelítése szerint a szoftverek komponenseinek interfészeit formálisan leírt, részletes és ellenőrizhető specifikációkkal együtt kell megtervezni. Ilyen módon egy osztály interfésze kiegészül a metódusokhoz tartozó elő- és utófeltételekkel, valamint az osztályhoz tartozó invariánssal. Szerződés alapú tervezés alapelvei: – Előfeltétel alapelv: Egy metódust meghívó kliensnek a hívás előtt meg kell bizonyosodnia arról, hogy az előfeltétel teljesül. – Utófeltétel alapelv: Egy metódusnak biztosítania kell, hogy amennyiben végrehajtásának kezdetén a hozzá társított előfeltétel teljesült, a végrehajtás végén a kapcsolódó utófeltétel is igaz legyen. – Osztályinvariáns alapelv: Az osztályinvariánsnak az objektum létrehozásától kezdve mindig igaznak kell lennie, minden, a kliensek számára elérhető metódus végrehajtása előtt és futtatása után.

Szerződéselemek megadása Java-ban: – Java Modeling Language (JML): interfészspecifikációs nyelv, amely nemcsak a szerződések leírására, de formális verifikációra is alkalmas. – Contracts for Java (cofoja): Google által fejlesztett ingyenes, nyílt forrású programkönyvtár, amelyben annotációk segítségével adhatjuk meg a szerződések elemeit.

Tag-interfész: egy másik interfészen vagy osztályon belül deklarált interfész.

Kívülről közvetlenül nem érhető el, csak a beágyazó interfész v. osztály nevével minősítve.

Ha intefészen belül deklaráljuk, akkor public. Viszont osztály tagjaként tetszőleges hozzáférési módosítója lehet.

Az interfész tagok automatikusan static módosítóval rendelkeznek (nem kell kiírni).

A lokális osztályok nem osztályszintűek, mert elérik a beágyazó blokk példányszintű tagjait.

A lokális osztályok a példányszintű tagosztályokhoz hasonlók.

Nem definiálhatnak static tagot (kivéve osztályszintű konstanst).

static metóduson belül a beágyazó osztálynak csak a static tagjait érik el. Mivel az interfész implicite static, blokkon belül nem lehet definiálni (nincs lokális interfész).

