1. **Jelölje be az összes igaz állítást! (max 4 pont)**

a) A Java tisztán objektum orientált nyelv

b) Javaban nem lehet struktúráltan programot írni

c) A Java fordító natív gépi kódra fordítja a forrásfájlokat

d) Egy Java osztályból csak 1 objektum készíthető

1. **Jelölje be az összes igaz állítást! (max 4 pont)**

a) Javaban megvalósítható a többszörös öröklődés

b) Az öröklődés csupán egy programozási technika és nem OOP alapelv

c) Mindig preferáljuk a meglévő kódbázis módosítását, testreszabását öröklődés helyett

d) Az öröklődés HAS-A kapcsolatot ír le

1. **Jelölje be az összes igaz állítást! (max 4 pont)**

a) Javaban minden osztály implicit az java.lang.Object osztályból származik

b) Az Object osztály ú.n. jelőlőosztály, nem tartalmaz metódusokat

c) A Serializable interfész implementálásával jelezhetjük a fordító számára, hogy az implementáció valamilyen natív nyelven less implementálva és nem Javaban

d) Az interfész implementáció jobban elősegíti az osztályok lazáncsatoltságát mint az öröklődés

1. **Mire van lehetőségünk az utód osztályban egy osztály örökítésekor? Jelölje be az összes helyes választ! (max 4 pont)**

a) Új adatokat deklarálhatunk.

b) Az ős osztály metódusait törölhetjük.

c) Az ős osztály adatait felüldefiniálhatjuk.

d) Az ős osztály metódusait felüldefiniálhatjuk.

**5. Jelölje be az összes igaz állítást! (max 4 pont)**

a) Öröklődés során a konstruktorokat is megörökli a leszármazott osztály

b) A konstrukor előtt nem szerepelhet final kulcsszó

c) Egy osztálynak pontosan 1 publikus és 1 privát konstruktora lehet

d) A példányszintű inicializációs blokk kiváltja a kontruktorok használatát

**6. Jelölje be az összes igaz állítást! (max 4 pont)**

a) Az osztályváltozó nem egy konkrét példányra, hanem az egész osztályra jellemző.

b) Az osztályváltozó osztályonként egyszer foglal memóriahelyet.

c) Egy példányszintű metódusban nem férünk hozzá az osztályváltozókhoz

d) Metódus nem lehet osztályszintű

**7. Jelölje meg az összes igaz állítást a következők közül! (max 4 pont)**

a) Az absztrakt osztályok kivételével minden osztályban deklarálni kell legalább egy konstruktort.

b) Egy osztálynak több, különböző szignatúrájú konstruktora is lehet.

c) Minden konstruktor nevének meg kell egyeznie az osztály nevével.

d) A konstruktor az objektum létrehozásakor fut le.

**9. Jelölje meg az összes igaz állítást a következők közül! (max 4 pont)**

a) Az interfész metódusai osztálymetódusok vagy példánymetódusok lehetnek.

b) Egy osztály akárhány interfészt implementálhat.

c) Az interfészt implementáló osztályban az interfésznek csak azokat a metódusait kell megírni,

amelyeket a program használni fog.

d) Egy osztály csak 1 interfészt implementálhat egyszerre

**10. Mi a polimorfizmus? Jelölje be az egyetlen jó választ! (max 4 pont)**

a) Ugyanarra az üzenetre különböző objektumok másképp reagálhatnak.

b) Sokféle objektumtípus (osztály) létezése.

c) Az egy objektumnak küldhető metódusok nagy száma.

d) Az egy objektumnak küldhető metódusok sokfélesége.

**11. Jelölje be az összes igaz állítást! (max 4 pont)**

a) A példányváltozót csak a példánymetódusok érik el.

b) A példányváltozót elérik a példánymetódusok és az osztálymetódusok is.

c) Az osztályváltozót csak az osztálymetódusok érik el.

d) Az osztályváltozót csak a példánymetódusok érik el.

**12. Jelölje be az összes igaz állítást! (max 4 pont)**

a) Egy referencia típusú változó azonosítójával közvetve hivatkozunk egy objektum

memóriahelyére.

b) A toString() metódust minden osztályban kötelezően meg kell írni.

c) Az equals() metódust minden osztályban kötelezően meg kell írni.

d) A System.out.println() metódus paramétere lehet egy objektum.

**13. Egy kivételkezelő szerkezetben az itt megadott blokkok szerepelnek, a következő sorrendben.**

**Jelölje meg az összes szintaktikailag helyes változatot! (max 4 pont)**

a) try – catch – catch – finally

b) catch – catch – finally

c) try – catch

d) finally – try

**14. Jelölje meg az összes igaz állítást a következők közül! (max 4 pont)**

a) A kivételt csak abban a metódusban lehet lekezelni, amelyben keletkezett.

b) A lekezeletlen kivétel a metódushívási láncon visszafelé kiszáll a programból.

c) Az ellenőrzött kivételek osztálya a RuntimeException vagy annak utódja.

d) Ha a kivételkezelésben van finally blokk, akkor az mindig végrehajtódik.

**Gyakorlati feladat1**

## **Space agency**

Valamikor a távoli jövőben...

A Naprendszert számos űreszköz járja, ezeket különböző szervezetek irányítják, megadják az aktuális céljukat. Az esetleges zavaró hatások miatt a SpaceAgency nyilvántartja ezeket, regisztrációs számuk és a kitűzött céljuk ismeretével. A célt mindig koordinátákkal adjuk meg, ez a Naphoz, mint origohoz rögzített, és egyes kitüntetett csillagok irányában felvett x, y és z irányokban vannak meghatározva. Az úticél módosítása is koordinátákban kerül megadásra, mindig a már megadott célponthoz képest a különbség kerül átadásra. Az űreszköz Satellite navigációs rendszere ennek alapján automatikusan irányítja önmagát.

## **Megvalósítás**

SpaceAgency osztály és feladatai: Regisztrálni lehet az útjukra indított eszközöket és azonosítójuk alapján ki is lehet keresni.

*public void registerSatellite(Satellite satellite)*

*public Satellite findSatelliteByRegisterIdent(String registerIdent)*

Satellite osztály és feladatai: A CelestialCoordinates immutable, azaz állapota nem módosítható. Az aktuális úticél új különbözeti koordináták megadásával frissíthető, ekkor a Satellite attribútuma új értéket vesz fel.

public Satellite(CelestialCoordinates destinationCoordinates, String registerIdent)

public void modifyDestination(CelestialCoordinates diff)

public String toString()

CelestialCoordinates osztály attribútumként a koordinátákat tartalmazza, ezek mind final változók, értéket a konstruktorban kapnak. Getter metódusok segítségével olvashatók.

## **Hibakezelés**

Üres String, mint paraméter nem fogadható el, továbbá a szatellit regisztrációnál nem kaphat null paramétert. Amennyiben a megadott azonosítóval nem található űreszköz, szintén kivételt várunk

**Gyakorlati feladat 2**

Készíts egy Pendrives osztályt, amely metódusaival pendrive-ok közül lehet kikeresni a megfelelőt. A feladat részeként készíts egy Pendrive osztályt is.

A Pendrive tagjai:

* attribútumai: name, capacity, price. A kapacitás egész szám Gb-ben megadott érték. Az ár egész szám forintban megadott érték.
* Legyenek getterei az attribútumokra.
* Legyen String toString() metódusa, amely egy Stringbe összefűzve adja vissza a pendrive adatait.
* Legyen void risePrice(int percent) metódusa, amely megadott százalékkal megemeli a pendrive árát.
* Legyen int comparePricePerCapacity(Pendrive) metódusa, amely összehasonlítja a példányt egy paraméterként kapott másik példánnyal az ár/kapacitás alapján. Az eredmény 1 legyen, ha az aktuális példány ár/kapacitása nagyobb, az eredmény -1 legyen ha a paraméterként kapott példányé nagyobb, és az eredmény 0 legyen, ha egyformák.
* Legyen egy boolean cheaperThan(Pendrive) metódusa, amely igazat ad ha a példány ára kisebb, mint a paraméterben kapott példány ára.

A Pendrives részletei: A metódusokat úgy implementáld, hogy nem hívod a Pendrive getPrice() metódusát.

* Legyen Pendrive best(List<Pendrive>) metódusa, amely a legjobb ár/kapacitás értékű pendrive-t adja vissza, azaz amelyiknél ez a legkisebb.
* Legyen Pendrive cheapest(List<Pendrive>) metódusa, amely a legolcsóbbat adja vissza.
* Legyen void risePriceWhereCapacity(List<Pendrive>, int percent, int capacity) metódusa, amely adott százalékkal megemeli azon pendrive-ok árát, amelynek a kapacitása a megadott értékkel megegyezik.