Programozási nyelvek II. JAVA EA+GY 1. gyakolat

EÖTVÖS LORÁND TUDOMÁNYEGYTEM
INFORMATIKAI KAR
PROGRAMOZÁSI NYELVEK ÉS FORDÍTÓPROGRAMOK TANSZÉK

2017/2018. őszi félév

Tartalom

Tartalom

Tartalom

Tartalom

Tartalom

Tartalom

Félév tematikája (1)

- 1. gyakorlat: program fordítása, vezérlési szerkezet, alapvető nyelvi elemek, programozási tételek Java-ban
- 2. gyakorlat: OOP alapelvek a gyakorlatban, ciklusok
- 3. gyakorlat: osztályok, konstruktor, láthatóság
- 4. gyakorlat: tömbök, aliasing, szivárogtatás
- 5. gyakorlat: kivételek alap szinten, fájlkezelés, JavaDoc
- 6. gyakorlat: felsorolási típus, túlterhelés, statikus adattagok és metódusok, default konstruktor
- 7. gyakorlat: gyakorlás

Félév tematikája (1)

- 1. gyakorlat: program fordítása, vezérlési szerkezet, alapvető nyelvi elemek, programozási tételek Java-ban
- 2. gyakorlat: OOP alapelvek a gyakorlatban, ciklusok
- 3. gyakorlat: osztályok, konstruktor, láthatóság
- 4. gyakorlat: tömbök, aliasing, szivárogtatás
- 5. gyakorlat: kivételek alap szinten, fájlkezelés, JavaDoc
- 6. gyakorlat: felsorolási típus, túlterhelés, statikus adattagok és metódusok, default konstruktor
- 7. gyakorlat: gyakorlás

Félév tematikája (1)

- 1. gyakorlat: program fordítása, vezérlési szerkezet, alapvető nyelvi elemek, programozási tételek Java-ban
- 2. gyakorlat: OOP alapelvek a gyakorlatban, ciklusok
- 3. gyakorlat: osztályok, konstruktor, láthatóság
- 4. gyakorlat: tömbök, aliasing, szivárogtatás
- 5. gyakorlat: kivételek alap szinten, fájlkezelés, JavaDoc
- 6. gyakorlat: felsorolási típus, túlterhelés, statikus adattagok és metódusok, default konstruktor
- 7. gyakorlat: gyakorlás

Félév tematikája (1)

- 1. gyakorlat: program fordítása, vezérlési szerkezet, alapvető nyelvi elemek, programozási tételek Java-ban
- 2. gyakorlat: OOP alapelvek a gyakorlatban, ciklusok
- 3. gyakorlat: osztályok, konstruktor, láthatóság
- 4. gyakorlat: tömbök, aliasing, szivárogtatás
- 5. gyakorlat: kivételek alap szinten, fájlkezelés, JavaDoc
- 6. gyakorlat: felsorolási típus, túlterhelés, statikus adattagok és metódusok, default konstruktor
- 7. gyakorlat: gyakorlás

Félév tematikája (1)

- 1. gyakorlat: program fordítása, vezérlési szerkezet, alapvető nyelvi elemek, programozási tételek Java-ban
- 2. gyakorlat: OOP alapelvek a gyakorlatban, ciklusok
- 3. gyakorlat: osztályok, konstruktor, láthatóság
- 4. gyakorlat: tömbök, aliasing, szivárogtatás
- 5. gyakorlat: kivételek alap szinten, fájlkezelés, JavaDoc
- 6. gyakorlat: felsorolási típus, túlterhelés, statikus adattagok és metódusok, default konstruktor
- 7. gyakorlat: gyakorlás

Félév tematikája (1)

- 1. gyakorlat: program fordítása, vezérlési szerkezet, alapvető nyelvi elemek, programozási tételek Java-ban
- 2. gyakorlat: OOP alapelvek a gyakorlatban, ciklusok
- 3. gyakorlat: osztályok, konstruktor, láthatóság
- 4. gyakorlat: tömbök, aliasing, szivárogtatás
- 5. gyakorlat: kivételek alap szinten, fájlkezelés, JavaDoc
- 6. gyakorlat: felsorolási típus, túlterhelés, statikus adattagok és metódusok, default konstruktor
- 7. gyakorlat: gyakorlás

Félév tematikája (1)

- 1. gyakorlat: program fordítása, vezérlési szerkezet, alapvető nyelvi elemek, programozási tételek Java-ban
- 2. gyakorlat: OOP alapelvek a gyakorlatban, ciklusok
- 3. gyakorlat: osztályok, konstruktor, láthatóság
- 4. gyakorlat: tömbök, aliasing, szivárogtatás
- 5. gyakorlat: kivételek alap szinten, fájlkezelés, JavaDoc
- 6. gyakorlat: felsorolási típus, túlterhelés, statikus adattagok és metódusok, default konstruktor
- 7. gyakorlat: gyakorlás

Félév tematikája (2)

- 8. gyakorlat: interface, collection, immutable design, csomagoló osztályok, boxing, generikusság
- 9. gyakorlat: öröklődés, felüldefiniálás, protected, super metódus és super konstruktor hívás
- 10. gyakorlat: absztrakt osztály, saját kivétel osztályok, throws, kivétel hierarhia
- 11. gyakorlat: equals, hashCode, comparable, clone
- 12. gyakorlat: lambda-kifejezések, Comparator
- 13. gyakorlat: gyakorlás

Félév tematikája (2)

- 8. gyakorlat: interface, collection, immutable design, csomagoló osztályok, boxing, generikusság
- 9. gyakorlat: öröklődés, felüldefiniálás, protected, super metódus és super konstruktor hívás
- 10. gyakorlat: absztrakt osztály, saját kivétel osztályok, throws, kivétel hierarhia
- 11. gyakorlat: equals, hashCode, comparable, clone
- 12. gyakorlat: lambda-kifejezések, Comparator
- 13. gyakorlat: gyakorlás

Félév tematikája (2)

- 8. gyakorlat: interface, collection, immutable design, csomagoló osztályok, boxing, generikusság
- 9. gyakorlat: öröklődés, felüldefiniálás, protected, super metódus és super konstruktor hívás
- 10. gyakorlat: absztrakt osztály, saját kivétel osztályok, throws, kivétel hierarhia
- 11. gyakorlat: equals, hashCode, comparable, clone
- 12. gyakorlat: lambda-kifejezések, Comparator
- 13. gyakorlat: gyakorlás

Félév tematikája (2)

- 8. gyakorlat: interface, collection, immutable design, csomagoló osztályok, boxing, generikusság
- 9. gyakorlat: öröklődés, felüldefiniálás, protected, super metódus és super konstruktor hívás
- 10. gyakorlat: absztrakt osztály, saját kivétel osztályok, throws, kivétel hierarhia
- 11. gyakorlat: equals, hashCode, comparable, clone
- 12. gyakorlat: lambda-kifejezések, Comparator
- 13. gyakorlat: gyakorlás

Félév tematikája (2)

- 8. gyakorlat: interface, collection, immutable design, csomagoló osztályok, boxing, generikusság
- 9. gyakorlat: öröklődés, felüldefiniálás, protected, super metódus és super konstruktor hívás
- 10. gyakorlat: absztrakt osztály, saját kivétel osztályok, throws, kivétel hierarhia
- 11. gyakorlat: equals, hashCode, comparable, clone
- 12. gyakorlat: lambda-kifejezések, Comparator
- 13. gyakorlat: gyakorlás

Félév tematikája (2)

- 8. gyakorlat: interface, collection, immutable design, csomagoló osztályok, boxing, generikusság
- 9. gyakorlat: öröklődés, felüldefiniálás, protected, super metódus és super konstruktor hívás
- 10. gyakorlat: absztrakt osztály, saját kivétel osztályok, throws, kivétel hierarhia
- 11. gyakorlat: equals, hashCode, comparable, clone
- 12. gyakorlat: lambda-kifejezések, Comparator
- 13. gyakorlat: gyakorlás

Követelmények (1)

- 2 Zárthelyi dolgozat
  - Félévközi zh
    - 2017. november 10. 14:00-20:00
    - 2017. november 17. 14:00-20:00 (pót)
  - Félévvégi zh (2 lehetőség)
    - 2017. december 22. 14:00-20:00 (csak első lehetőségként)
    - 2018. január 12. 14:00-20:00 (csak első lehetőségként)
    - 2018. január 29. 14:00-20:00 (csak második lehetőségként)
- 2 beadandó
  - a ZH-k előtt lesz kiírva
  - kötelező előfeltétele a ZH-nak
  - az addigi tananyagot kéri számon
  - határidők: 2017. október 31. és 2017. december 10.

Követelmények (1)

- 2 Zárthelyi dolgozat
  - Félévközi zh
    - 2017. november 10. 14:00-20:00
    - 2017. november 17. 14:00-20:00 (pót)
  - Félévvégi zh (2 lehetőség)
    - 2017. december 22. 14:00-20:00 (csak első lehetőségként)
    - 2018. január 12. 14:00-20:00 (csak első lehetőségként)
    - 2018. január 29. 14:00-20:00 (csak második lehetőségként)
- 2 beadandó
  - a ZH-k előtt lesz kiírva
  - kötelező előfeltétele a ZH-nak
  - az addigi tananyagot kéri számon
  - határidők: 2017. október 31. és 2017. december 10.

Követelmények (1)

- 2 Zárthelyi dolgozat
  - Félévközi zh
    - 2017. november 10. 14:00-20:00
    - 2017. november 17. 14:00-20:00 (pót)
  - Félévvégi zh (2 lehetőség)
    - 2017. december 22. 14:00-20:00 (csak első lehetőségként)
    - 2018. január 12. 14:00-20:00 (csak első lehetőségként)
    - 2018. január 29. 14:00-20:00 (csak második lehetőségként)
- 2 beadandó
  - a ZH-k előtt lesz kiírva
  - kötelező előfeltétele a ZH-nak
  - az addigi tananyagot kéri számon
  - határidők: 2017. október 31. és 2017. december 10.

Követelmények (2)

- Elméleti jegy
  - 10 +/- elméleti kérdésekből a gyakorlatokon
  - 2 zárthelyi dolgozaton elméleti kérdések, 14 és 15 pontért
  - ponthatárok: 20, 25, 30, 35
- Végső jegy
  - a két gépes zárthelyi és az elméleti jegy átlaga
  - amennyiben mindhárom legalább elégséges

Követelmények (2)

- Elméleti jegy
  - 10 +/- elméleti kérdésekből a gyakorlatokon
  - 2 zárthelyi dolgozaton elméleti kérdések, 14 és 15 pontért
  - ponthatárok: 20, 25, 30, 35
- Végső jegy
  - a két gépes zárthelyi és az elméleti jegy átlaga
  - amennyiben mindhárom legalább elégséges

Követelmények (3)

- maximum 3 (nagyon indokolt esetben 4) gyakorlatról lehet hiányozni
- akinek az első zárhelyi dolgozata nem sikerült (a pót sem), vagy valamelyik beadandóját nem adta be, az már nem végezheti el a tárgyat
- a beadott szoftverek plágium-ellenőrzésen mennek keresztül, akit másoláson kapunk (és az is, akiről másolta) nem végezheti el a tantárgyat

Követelmények (3)

- maximum 3 (nagyon indokolt esetben 4) gyakorlatról lehet hiányozni
- akinek az első zárhelyi dolgozata nem sikerült (a pót sem), vagy valamelyik beadandóját nem adta be, az már nem végezheti el a tárgyat
- a beadott szoftverek plágium-ellenőrzésen mennek keresztül, akit másoláson kapunk (és az is, akiről másolta) nem végezheti el a tantárgyat

Követelmények (3)

- maximum 3 (nagyon indokolt esetben 4) gyakorlatról lehet hiányozni
- akinek az első zárhelyi dolgozata nem sikerült (a pót sem), vagy valamelyik beadandóját nem adta be, az már nem végezheti el a tárgyat
- a beadott szoftverek plágium-ellenőrzésen mennek keresztül, akit másoláson kapunk (és az is, akiről másolta) nem végezheti el a tantárgyat

Történet, jellemzők

- James Gosling kezdte fejleszteni a Sun Microsystemsnél a 90-es évek elején
- Első kiadás 1995-ös, azóta sok minden változott, bővült a nyelv
- A Java 2 Platform, Sandard Edition (SE) 5.0 2004-es
- Objektumorientált nyelv (primitív típusokon kívül minden az Object ősosztály leszármazottja)
- C++ szerű szintaxis
- a fordító bájtkódra fordít, a futtatásért a Java Virtual Machine (JVM) felel
- a bájtkód hordozható platformok között (Windows, Unix, Linux, Macintosh), ami nagyon nagy előny

Történet, jellemzők

- James Gosling kezdte fejleszteni a Sun Microsystemsnél a 90-es évek elején
- Első kiadás 1995-ös, azóta sok minden változott, bővült a nyelv
- A Java 2 Platform, Sandard Edition (SE) 5.0 2004-es
- Objektumorientált nyelv (primitív típusokon kívül minden az Object ősosztály leszármazottja)
- C++ szerű szintaxis
- a fordító bájtkódra fordít, a futtatásért a Java Virtual Machine (JVM) felel
- a bájtkód hordozható platformok között (Windows, Unix, Linux, Macintosh), ami nagyon nagy előny

Történet, jellemzők

- James Gosling kezdte fejleszteni a Sun Microsystemsnél a 90-es évek elején
- Első kiadás 1995-ös, azóta sok minden változott, bővült a nyelv
- A Java 2 Platform, Sandard Edition (SE) 5.0 2004-es
- Objektumorientált nyelv (primitív típusokon kívül minden az Object ősosztály leszármazottja)
- C++ szerű szintaxis
- a fordító bájtkódra fordít, a futtatásért a Java Virtual Machine (JVM) felel
- a bájtkód hordozható platformok között (Windows, Unix, Linux, Macintosh), ami nagyon nagy előny

Történet, jellemzők

- James Gosling kezdte fejleszteni a Sun Microsystemsnél a 90-es évek elején
- Első kiadás 1995-ös, azóta sok minden változott, bővült a nyelv
- A Java 2 Platform, Sandard Edition (SE) 5.0 2004-es
- Objektumorientált nyelv (primitív típusokon kívül minden az Object ősosztály leszármazottja)
- C++ szerű szintaxis
- a fordító bájtkódra fordít, a futtatásért a Java Virtual Machine (JVM) felel
- a bájtkód hordozható platformok között (Windows, Unix, Linux, Macintosh), ami nagyon nagy előny

Történet, jellemzők

- James Gosling kezdte fejleszteni a Sun Microsystemsnél a 90-es évek elején
- Első kiadás 1995-ös, azóta sok minden változott, bővült a nyelv
- A Java 2 Platform, Sandard Edition (SE) 5.0 2004-es
- Objektumorientált nyelv (primitív típusokon kívül minden az Object ősosztály leszármazottja)
- C++ szerű szintaxis
- a fordító bájtkódra fordít, a futtatásért a Java Virtual Machine (JVM) felel
- a bájtkód hordozható platformok között (Windows, Unix, Linux, Macintosh), ami nagyon nagy előny

Történet, jellemzők

- James Gosling kezdte fejleszteni a Sun Microsystemsnél a 90-es évek elején
- Első kiadás 1995-ös, azóta sok minden változott, bővült a nyelv
- A Java 2 Platform, Sandard Edition (SE) 5.0 2004-es
- Objektumorientált nyelv (primitív típusokon kívül minden az Object ősosztály leszármazottja)
- C++ szerű szintaxis
- a fordító bájtkódra fordít, a futtatásért a Java Virtual Machine (JVM) felel
- a bájtkód hordozható platformok között (Windows, Unix, Linux, Macintosh), ami nagyon nagy előny

Történet, jellemzők

- James Gosling kezdte fejleszteni a Sun Microsystemsnél a 90-es évek elején
- Első kiadás 1995-ös, azóta sok minden változott, bővült a nyelv
- A Java 2 Platform, Sandard Edition (SE) 5.0 2004-es
- Objektumorientált nyelv (primitív típusokon kívül minden az Object ősosztály leszármazottja)
- C++ szerű szintaxis
- a fordító bájtkódra fordít, a futtatásért a Java Virtual Machine (JVM) felel
- a bájtkód hordozható platformok között (Windows, Unix, Linux, Macintosh), ami nagyon nagy előny

Java program

- HelloWorld:

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

- Egy java fájl tartalmaz egy azonos nevű osztályt (a HelloWorld.java fájlban van egy HelloWorld osztály)
- A függvények osztálydefiníción belül fordulhatnak elő
- Fordításkor a HelloWorld.java-ból készül egy HelloWorld.class fájl ('javac' paranccsal)
- Ha az osztálynak van 'main()' metódusa (belépési pontja), akkor végrehajtható (a 'java' paranccsal)
- Az osztályok úgynevezett 'package'-ekbe (csomagokba) szerveződnek

Java program

- HelloWorld:

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

- Egy java fájl tartalmaz egy azonos nevű osztályt (a HelloWorld.java fájlban van egy HelloWorld osztály)
- A függvények osztálydefiníción belül fordulhatnak elő
- Fordításkor a HelloWorld.java-ból készül egy HelloWorld.class fájl (’javac’ paranccsal)
- Ha az osztálynak van ’main()’ metódusa (belépési pontja), akkor végrehajtható (a ’java’ paranccsal)
- Az osztályok úgynevezett ’package’-ekbe (csomagokba) szerveződnek

Java program

- HelloWorld:

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

- Egy java fájl tartalmaz egy azonos nevű osztályt (a HelloWorld.java fájlban van egy HelloWorld osztály)
- A függvények osztálydefiníción belül fordulhatnak elő
- Fordításkor a HelloWorld.java-ból készül egy HelloWorld.class fájl (’javac’ paranccsal)
- Ha az osztálynak van ’main()’ metódusa (belépési pontja), akkor végrehajtható (a ’java’ paranccsal)
- Az osztályok úgynevezett ’package’-ekbe (csomagokba) szerveződnek

Java program

- HelloWorld:

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

- Egy java fájl tartalmaz egy azonos nevű osztályt (a HelloWorld.java fájlban van egy HelloWorld osztály)
- A függvények osztálydefiníción belül fordulhatnak elő
- Fordításkor a HelloWorld.java-ból készül egy HelloWorld.class fájl (’javac’ paranccsal)
- Ha az osztálynak van ’main()’ metódusa (belépési pontja), akkor végrehajtható (a ’java’ paranccsal)
- Az osztályok úgynevezett ’package’-ekbe (csomagokba) szerveződnek

Java program

- HelloWorld:

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

- Egy java fájl tartalmaz egy azonos nevű osztályt (a HelloWorld.java fájlban van egy HelloWorld osztály)
- A függvények osztálydefiníción belül fordulhatnak elő
- Fordításkor a HelloWorld.java-ból készül egy HelloWorld.class fájl (’javac’ paranccsal)
- Ha az osztálynak van ’main()’ metódusa (belépési pontja), akkor végrehajtható (a ’java’ paranccsal)
- Az osztályok úgynevezett ’package’-ekbe (csomagokba) szerveződnek

Fordítás, futtatás (parancssor)(1)

- Fordítás
  - javac paranccsal történik
  - -d argumentummal megadható, hogy milyen könyvtárba kerüljenek a class fájlok
  - Példa: javac HelloWorld.java (abszolút vagy relatív elérési út)
- Futtatás
  - java parancs
  - -cp argumentummal beállítható, hogy honnan vegye a class fájlokat
  - Példa: java HelloWorld

Fordítás, futtatás (parancssor)(1)

- Fordítás
  - javac paranccsal történik
  - -d argumentummal megadható, hogy milyen könyvtárba kerüljenek a class fájlok
  - Példa: javac HelloWorld.java (abszolút vagy relatív elérési út)
- Futtatás
  - java parancs
  - -cp argumentummal beállítható, hogy honnan vegye a class fájlokat
  - Példa: java HelloWorld

Fordítás, futtatás (parancssor)(2)

- Ha nem ismertek a parancsok be kell állítani
  Windows:

```
set PATH=%PATH%;c:\Program Files\Java\jdk1.8.0_144\bin\
```

  Linux:

```
gedit /etc/profile
```

```
export PATH=$PATH:/usr/java/jdk1.8.0_144/bin/
```

Kódolási konvenció, dokumentáció

- Kódolási konvenció
  - Fontos betartani az előírt kódolási konvenciókat (részletek később).
  - Fő szempontok: olvashatóság, egyértelműség, értelmezhetőség
- Dokumentáció
  - Az előadás honlapja
  - Java tutorial
  - Java API doc

Típusok, konverziók

- Típusok
  - primitívek: byte, short, int, long, float, double, char, boolean
  - objektumok
  - tömbök
  - Változó definiálás

```
boolean b = true;
```

- Az operátorok más nyelvből megszokottak (Hivatalos operátor lista)

Típusok, konverziók

- Típusok
  - primitívek: byte, short, int, long, float, double, char, boolean
  - objektumok
  - tömbök
  - Változó definiálás

```
boolean b = true;
```

- Az operátorok más nyelvből megszokottak (Hivatalos operátor lista)

Típusok, konverziók

- Típusok
  - primitívek: byte, short, int, long, float, double, char, boolean
  - objektumok
  - tömbök
  - Változó definiálás

```
boolean b = true;
```

- Az operátorok más nyelvből megszokottak (Hivatalos operátor lista)

Típusok, konverziók

- Típusok
  - primitívek: byte, short, int, long, float, double, char, boolean
  - objektumok
  - tömbök
  - Változó definiálás

```
boolean b = true;
```

- Az operátorok más nyelvből megszokottak (Hivatalos operátor lista)

Elágazások

- Kétirányú elágazás

```
if ( i % 5 == 0 ) {
   System.out.println( i+" 5-el osztható" );
}
else {  System.out.println( i+" 5-el nem osztható" ); }
```

- Többirányú elágazás (Csak primitív, felsorolási és String típusra működik.)

```
switch ( i % 4 ){
    case 1: System.out.println( i+" 4-es maradéka 1" );
            break;
    case 2: System.out.println( i+" 4-es maradéka 2" );
            break;
    case 3: System.out.println( i+" 4-es maradéka 3" );
            break;
    default: System.out.println( i+" 4-el osztható" );
             break;
}
```

Elágazások

- Kétirányú elágazás

```
if ( i % 5 == 0 ) {
   System.out.println( i+" 5-el osztható" );
}
else {  System.out.println( i+" 5-el nem osztható" ); }
```

- Többirányú elágazás (Csak primitív, felsorolási és String típusra működik.)

```
switch ( i % 4 ){
    case 1: System.out.println( i+" 4-es maradéka 1" );
            break;
    case 2: System.out.println( i+" 4-es maradéka 2" );
            break;
    case 3: System.out.println( i+" 4-es maradéka 3" );
            break;
    default: System.out.println( i+" 4-el osztható" );
             break;
}
```

Ciklusok

- Elöl tesztelő ciklus

```
int i =  0;
while ( i < 3 ){
    System.out.println( i );
}
```

- Hátul tesztelő ciklus

```
int i = 0;
do {
   i++;
} while ( i < 5 );
```

- Léptető ciklus

```
for ( int i = 0; i < 5; i += 2){
    System.out.println( i );
}
```

- Iteráló ciklus

```
List<String> args = new ArrayList<>();
for (String arg : args){ System.out.println( args ); }
```

Ciklusok

- Elöl tesztelő ciklus

```
int i =  0;
while ( i < 3 ){
    System.out.println( i );
}
```

- Hátul tesztelő ciklus

```
int i = 0;
do {
   i++;
} while ( i < 5 );
```

- Léptető ciklus

```
for ( int i = 0; i < 5; i += 2){
    System.out.println( i );
}
```

- Iteráló ciklus

```
List<String> args = new ArrayList<>();
for (String arg : args){ System.out.println( args ); }
```

Ciklusok

- Elöl tesztelő ciklus

```
int i =  0;
while ( i < 3 ){
    System.out.println( i );
}
```

- Hátul tesztelő ciklus

```
int i = 0;
do {
   i++;
} while ( i < 5 );
```

- Léptető ciklus

```
for ( int i = 0; i < 5; i += 2){
    System.out.println( i );
}
```

- Iteráló ciklus

```
List<String> args = new ArrayList<>();
for (String arg : args){ System.out.println( args ); }
```

Ciklusok

- Elöl tesztelő ciklus

```
int i =  0;
while ( i < 3 ){
    System.out.println( i );
}
```

- Hátul tesztelő ciklus

```
int i = 0;
do {
   i++;
} while ( i < 5 );
```

- Léptető ciklus

```
for ( int i = 0; i < 5; i += 2){
    System.out.println( i );
}
```

- Iteráló ciklus

```
List<String> args = new ArrayList<>();
for (String arg : args){ System.out.println( args ); }
```

Ugró utasítások

- continue
  - Továbblép a ciklus következő iterációjára
- break
  - Kilép a ciklusból és a ciklus utáni első vezérlési szerkezetet hajtja végre
- return
  - Visszatérés egy függvényből vagy eljárásból

Ugró utasítások

- continue
  - Továbblép a ciklus következő iterációjára
- break
  - Kilép a ciklusból és a ciklus utáni első vezérlési szerkezetet hajtja végre
- return
  - Visszatérés egy függvényből vagy eljárásból

Ugró utasítások

- continue
  - Továbblép a ciklus következő iterációjára
- break
  - Kilép a ciklusból és a ciklus utáni első vezérlési szerkezetet hajtja végre
- return
  - Visszatérés egy függvényből vagy eljárásból

1. gyakorló feladat

Töltse le az alábbi forrásfájlt és fordítsa le.
Elemezze a működését, figyelje meg az egyes nyelvi elemeket.

2. gyakorló feladat

Egészítse ki a forrást egy olyan programrészlettel, amely egy tetszőleges számig csak a 3-mal és 5-tel osztható számok összegét számolja ki.
(A megoldáshoz használjon ciklust és feltételes utasítást.)

3. gyakorló feladat

Keresse meg a hibát az alábbi forrásban, és javítsa ki, hogy a forrás forduljon és helyesen fusson le.
A program ismételten növeli egy változó értékét majd kiírja azt.

4. gyakorló feladat

Töltse le az alábbi forrásfájlt.
A forrásfájl egy egyszerű órát utánoz. A ciklus változó egy ezredmásodperc számláló, amely az órában a ezredmásodpercek számáig ér. A ciklus törzse pedig percszámláló, vagyis rendszeres időközönként megnöveli az értékét. Végül kiírja az órában lévő percek számát.
Futtassa le a forrást és nézze meg a kimenetet. Ha hibás keresse meg a hiba okát.

Köszönöm a figyelmet!