ΤΕΧΝΙΚΕΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Επεξεργασία αλφαριθμητικών

Στατικές μέθοδοι και μεταβλητές

Εσωτερικές κλάσεις

STRING PROCESSING

Strings

- Η επεξεργασία αλφαριθμητικών είναι πολύ σημαντική για πολλές εφαρμογές. Θα δούμε μερικές χρήσιμες εντολές
- Σε όλες τις εντολές για επεξεργασία των Strings δεν πρέπει να ξεχνάμε ότι τα Strings είναι immutable objects
  - Οι μέθοδοι που καλεί μια μεταβλητή String δεν μπορούν να αλλάξουν την μεταβλητή, μόνο να επιστρέψουν ένα νέο String.

toLowerCase, trim

- Οι παρακάτω εντολές είναι χρήσιμες για να κανονικοποιούμε το String
  - toLowerCase(): μετατρέπει όλους τους χαρακτήρες ενός String σε μικρά γράμματα.
  - trim(): αφαιρεί λευκούς χαρακτήρες (κενά, tabs, αλλαγή γραμμής) από την αρχή και το τέλος

- Χρήσιμες εντολές όταν κάνουμε συγκρίσεις μεταξύ Strings και θέλουμε να τα φέρουμε σε κοινή μορφή.

Παράδειγμα

```
public class StringTest1
{
        public static void main(String args[]){
                String s1 = "this is a sentence ";
                String s2 = "This is a sentence";

                System.out.println(s1);
                System.out.println(s2);
                System.out.println(s1.equals(s2));

                s1 = s1.trim();
                s2 = s2.toLowerCase();
                System.out.println(s1);
                System.out.println(s2);
                System.out.println(s1.equals(s2));
        }
}
```

Για να αποφεύγονται κενά στην αρχή η στο τέλος

Χρήσιμη εντολή για συγκρίσεις λέξεων, για να μην εξαρτόμαστε αν η λέξη είναι σε μικρά ή κεφαλαία

Πρέπει **πάντα** να γίνεται ξανά ανάθεση στη μεταβλητή. Η εντολή `s2.toLowerCase();` δεν αλλάζει το s2 επιστρέφει το αλλαγμένο String.

split

- Η εντολή split είναι χρήσιμη για να σπάμε ένα String σε πεδία που διαχωρίζονται από ένα συγκεκριμένο string (delimeter)
  - Όρισμα: το string ως προς το οποίο θέλουμε να σπάσουμε το κείμενο.
  - Επιστρέφει: πίνακα String[] με τα πεδία που δημιουργήθηκαν.

Παράδειγμα: από το String:
"Student: Bob Marley AM: 111"
θέλουμε το όνομα του φοιτητή και το ΑΜ του

```
class SplitTest1{
  public static void main(String args[]){
    String s = "Student: Bob Marley\tAM: 111";
    System.out.println(s);

    String fields[] = s.split("\t");

    String studentFields[] = fields[0].split(":");
    String studentName = studentFields[1].trim();

    String AMFields[] = fields[1].split(":");
    int studentAM = Integer.parseInt(AMFields[1].trim());

    System.out.println(studentName + "\t" + studentAM);
  }
}
```

Split πρώτα ως προς "\t" και μετά ως προς ":"

Χρήση της trim

replace

- Η εντολή είναι χρήσιμη αν θέλουμε να αλλάξουμε κάπως το String
  - replace(String before, String after): αντικαθιστά το before με το after και επιστρέφει το αλλαγμένο String

Παράδειγμα

```
class ReplaceTest1
{
    public static void main(String[] args){
        String s1 = "Is this a greek question?";
        System.out.println("Before:" + s1);
        s1 = s1.replace("?",";");
        System.out.println("After:" + s1);

        String s2 = "This is not a question?";
        System.out.println("Before:" + s2);
        s2 = s2.replace("?", "");
        System.out.println("After:" + s2);

        String s3 = "20-5-2013";
        System.out.println("Before:" + s3);
        s3 = s3.replace("-","/");
        System.out.println("After:" + s3);

        }
}
```

Αντικαθιστά το “?” με “;”

Σβήνει το “?”

Αντικαθιστά όλα τα “-” με “/”

Split και Replace

- Υπάρχουν περιπτώσεις που θέλουμε να σπάσουμε ή να αντικαταστήσουμε με βάση κάτι πιο περίπλοκο από ένα String
  - Π.χ., θέλουμε να σπάσουμε ένα String ως προς tabs ή κενά
  - Π.χ., θέλουμε να σβήσουμε οτιδήποτε είναι ερωτηματικό, ελληνικό ή αγγλικό
  - Π.χ., θέλουμε να σβήσουμε τις τελείες αλλά μόνο αν είναι στο τέλος του String.
- Για να προσδιορίσουμε τέτοιες περίπλοκες περιπτώσεις χρησιμοποιούμε κανονικές εκφράσεις (regular expressions)

Regular Expressions

- Ένας τρόπος να περιγράφουμε Strings που έχουν ακολουθούν ένα κοινό μοτίβο
  - Έχετε ήδη χρησιμοποιήσει κανονικές εκφράσεις. Όταν γράφετε "`ls *.txt`" το "`*.txt`" είναι μια κανονική έκφραση που περιγράφει όλα τα Strings που τελειώνουν σε "`.txt`"
- Μια κανονική έκφραση λέμε ότι ταιριάζει (matches) με ένα string όταν το string περιγράφεται από το γενικό μοτίβο της κανονικής έκφρασης.

Κανονικές Εκφράσεις στη Java

- Μπορείτε να διαβάσετε μια περίληψη στη σελίδα της Oracle
- Οι κανονικές εκφράσεις μπορούν να περιγράψουν πολλά πράγματα. Εμείς θα χρησιμοποιήσουμε κάποιες απλές εκφράσεις.
- Παραδείγματα:
  - [abc]: ταιριάζει με a ή b ή c
  - ^a : ταιριάζει με ένα a που εμφανίζεται στην αρχή του String.
  - a$: ταιριάζει με ένα a που εμφανίζεται στο τέλος του String
  - \s ή \p{Space}: ταιριάζει με οποιοδήποτε white space (κενό, tab, αλλαγή γραμμής)
  - \p{Punct}: ταιριάζει όλα τα σημεία στίξης
  - a*: ταιριάζει 0 ή παραπάνω εμφανίσεις του a
  - a+: ταιριάζει 1 ή παραπάνω εμφανίσεις του a

- Για να χρησιμοποιήσουμε τις κανονικές εκφράσεις τις μετατρέπουμε σε ένα string που δίνεται ως όρισμα στην split η την replaceAll.
  - Π.χ. `"[abc]"`, `"^a"`, `"a$"`, `"\\s"`, `"\\p{Space}"`, `"\\p{Punct}"`
    - Χρειαζόμαστε το "\\" ώστε να βάλουμε το \ μέσα στο string.

Παρένθεση

- Ο χαρακτήρας **** λέγεται escape character
  - Όταν τον συνδυάζουμε με άλλους χαρακτήρες παίρνει διαφορετικό νόημα όταν είμαστε μέσα σε String
    - **\n**: αλλαγή γραμμής
    - **\t**: tab
    - **\“**: ο χαρακτήρας **“**
    - ****: ο χαρακτήρας ****

Παράδειγμα

```
class SplitTest2
{
    public static void main(String args[]){
        String s1 = "sentense 1\tsentence 2";
        String[] tokens = s1.split("[\t ]");
        for (String t: tokens){
            System.out.println(t);
        }
        tokens = s1.split("\\s");
        for (String t: tokens){
            System.out.println(t);
        }

        String s2 =  "To be or not to be? This is the question. The
question we must face";
        String[] sentences = s2.split("[?.]");
        for (String s: sentences){
            System.out.println(s.trim());
        }
    }
}
```

Split στο tab και το κενό

Split σε οποιοδήποτε white space

Split στο ερωτηματικό και την τελεία

Σβήνει τα κενά στην αρχή και το τέλος των προτάσεων

Παράδειγμα

Για να χρησιμοποιήσουμε την κανονική έκφραση χρειαζόμαστε την εντολή replaceAll

```
class ReplaceTest2
{
    public static void main(String args[]){
        String s = "The cost is 99.99 dollars.";
        System.out.println(s);
        s = s.replaceAll("[.]$","");
        System.out.println(s);

        s = "\"Quoted (\"quote\") text\"";
        System.out.println(s);
        s = s.replaceAll("^\"","");
        s = s.replaceAll("\"$","");
        System.out.println(s);

        s = "What?Yes!No...";
        System.out.println(s);
        s = s.replaceAll("[.!?]"," ");
        //s = s.replaceAll("\\p{Punct}"," "); // εναλλακτικά
        System.out.println(s);

        s = "Space: Tab:\t:End";
        System.out.println(s);
        s = s.replaceAll("\\p{Space}","");
        System.out.println(s);
    }
}
```

Σβήνει την τελεία στο τέλος του String

Σβήνει το “ στην αρχή του String

Σβήνει το ” στο τέλος του String

Αντικαθιστά τελεία, θαυμαστικό και ερωτηματικό με κενό.

Εναλλακτικός τρόπος να αντικαταστήσουμε τα σημεία στίξεως με κενά.

Σβήνει τους whitespace χαρακτήρες

```
class ReplaceTest3
{
    public static void main(String args[]){
        String s = "Hello...";
        s = s.replaceAll("[.]$","");
        System.out.println(s);

        s = s.replaceAll("[.]*$","");
        System.out.println(s);

        s = "\" Quoted (\"quote\") text \"";
        String[] words = s
                .toLowerCase()
                .replaceAll("^\"","")
                .replaceAll("\"$","")
                .trim()
                .split();
        System.out.println(s);
    }
}
```

Τι θα τυπώσει?

Σβήνει μία τελεία από το τέλος του String

Πως μπορούμε να σβήσουμε όλες τις τελείες?

Θέλουμε από το s να αφαιρέσουμε τα αρχικά και τελικά " να αφαιρέσουμε αρχικά και τελικά κενά να μετατρέψουμε τα γράμματα σε μικρά και να το σπάσουμε σε λέξεις

Για να μην κάνουμε συνεχείς αναθέσεις των αποτελεσμάτων των μεθόδων βολεύει να κάνουμε αλυσιδωτές κλήσεις των μεθόδων.

StringTokenizer

- Η διαδικασία του να σπάμε ένα string σε κομμάτια που χωρίζονται με κενά λέγεται tokenization και τα κομμάτια tokens.
- Η κλάση StringTokenizer κάνει και το tokenization και μας επιτρέπει να διατρέχουμε τα tokens
  - StringTokenizer st = new StringTokenizer(s): Δημιουργέι ένα tokenizer για το String s, με διαχωριστικό (delimeter) τους λευκούς χαρακτήρες (\s)
  - nextToken(): επιστρέφει το επόμενο token
  - hasMoreTokens(): μας λέει αν έχουμε άλλα tokens
- Θα μπορούσαμε να χρησιμοποιήσουμε και την split αλλά η StringTokenizer χειρίζεται αυτόματα τις διάφορες περιπτώσεις με white space
  - Π.χ. πολλαπλά κενά

Παράδειγμα

```
import java.util.StringTokenizer;

class StringTokenizerTest
{
    public static void main(String args[]){
        String s = "Line with tab\t and space";
        System.out.println(s);

        System.out.println("Split tokenization");
        String[] tokens1 = s.split("\\s");
        for (String t: tokens1){
            System.out.println("-"+t+"-");
        }

        System.out.println("StringTokenizer tokenization");
        StringTokenizer tokens2  = new StringTokenizer(s);
        while (tokens2.hasMoreTokens()){
            System.out.println("-"+tokens2.nextToken()+"-");
        }
    }
}
```

Split σε κενό και tab

Δημιουργεί κενό token όταν βρει το "\t "

Δεν δημιουργεί κενό token όταν βρει το "\t "

Παράδειγμα

```
import java.util.StringTokenizer;

class StringTokenizerTest
{
    public static void main(String args[]){
        String s = "Line with tab\t and space";
        System.out.println(s);

        System.out.println("Split tokenization");
        String[] tokens1 = s.split("\\s+");
        for (String t: tokens1){
            System.out.println("-"+t+"-");
        }

        System.out.println("StringTokenizer tokenization");
        StringTokenizer tokens2  = new StringTokenizer(s);
        while (tokens2.hasMoreTokens()){
            System.out.println("-"+tokens2.nextToken()+"-");
        }
    }
}
```

Split σε τουλάχιστον ένα κενό ή tab

Δεν δημιουργεί κενό token

StringTokenizer

- Μπρούμε να κάνουμε tokenization και με διαφορετικά διαχωριστικά. Αυτά τα προσδιορίζουμε στον constructor.
  - `StringTokenizer st =`

    `new StringTokenizer(s,".?!");`
    - Δημιουργέι ένα tokenizer για το String s, με διαχωριστικό (delimeter) την τελεία, το ερωτηματικό και το θαυμαστικό.

```
import java.util.StringTokenizer;

class StringTokenizerTest2
{
   public static void main(String args[]){
       String s = "The first sentence. The second! Third? And,
finally, the last one.";
       System.out.println(s);
       StringTokenizer tokens  = new StringTokenizer(s,".?!");
       System.out.println("Tokenization:");
       while (tokens.hasMoreTokens()){
          System.out.println(tokens.nextToken().trim());
       }
   }
}
```

StringBuilder

- Τα Strings είναι immutable objects. Αυτό σημαίνει ότι για να αλλάξουμε ένα String πρέπει να το ξανα-δημιουργήσουμε και να το αντιγράψουμε
- Για τέτοιου είδους αλλαγές είναι καλύτερα να χρησιμοποιούμε την κλάση StringBuilder
  - append: προσθέτει ένα String στο τέλος του υπάρχοντος. Παίρνει σαν όρισμα String ή οποιοδήποτε πρωταρχικό τύπο. Αν πάρει όρισμα κάποιο αντικείμενο καλείται αυτόματα η μέθοδος toString του αντικειμένου.
  - toString(): επιστρέφει το τελικό String

- Πολύ βολικό για να δημιουργούμε String συνενώνοντας πολλαπλά Strings.

```
import java.lang.StringBuilder;

class StringBuilderTest
{
        public static void main(String[] args){
                int N = 100000;

                String s = "";
                for (int i = 0; i < 100000; i ++){
                        s = s + " " +i;
                }
                System.out.println(s);


                StringBuilder sb = new StringBuilder();
                for (int i = 0; i < 100000; i ++){
                        sb.append(" " +i);
                }
                System.out.println(sb.toString());
        }
}
```

Θέλουμε να δημιουργήσουμε ένα String με τους αριθμούς από το 1 ως το N

Ο μπλε κώδικας είναι **πολύ** πιο γρήγορος από τον πράσινο
Ο πράσινος αντιγράφει το String N φορές

```
import java.lang.StringBuilder;

class StringBuilderTest2
{
	public static void main(String[] args){
		StringBuilder sb = new StringBuilder();
		for (int i = 0; i < 10; i ++){
			Person p = new Person("Some Person",i);
			sb.append(p+"\n");
		}
		String s = sb.toString();
		System.out.println(s);
	}
}
```

Καλείται η μέθοδος toString της Person και συνενώνεται στο τέλος του υπάρχοντος String

ΠΑΡΑΔΕΙΓΜΑ

Αρχεία – Επεξεργασία αλφαριθμητικών - Δομές

Παράδειγμα

- Έχουμε ένα αρχείο studentNames.txt με τα ΑΜ και τα ονόματα των φοιτητών (tab-separated) και ένα αρχείο studentGrades.txt με τα ΑΜ και βαθμό (για κάποια μαθήματα – ένα μάθημα ανά γραμμή). Τυπώστε σε ένα αρχείο ΑΜ, όνομα, βαθμό.

```
import java.util.Scanner;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.io.FileOutputStream;

import java.util.HashMap;

class Join
{
    public static void main(String[] args){
        Scanner nameInputStream = null;
        Scanner gradesInputStream = null;
        PrintWriter outputStream = null;

        try
        {
           nameInputStream = new Scanner(
                             new FileInputStream("studentNames.txt"));
           gradesInputStream = new Scanner(
                             new FileInputStream("studentGrades.txt"));
           outputStream = new PrintWriter(
                             new FileOutputStream("studentNamesGrades.txt"));
        }
        catch(FileNotFoundException e)
        {
           System.out.println("Problem opening files.");
           System.exit(0);
        }
```

Άνοιγμα των αρχείων εισόδου για διάβασμα και του αρχείου εξόδου για γράψιμο

Συνέχεια στην επόμενη

Συνέχεια από την προηγούμενη

```
        HashMap<Integer,String> namesHash = new HashMap<Integer,String>();
        while (nameInputStream.hasNextLine( ))
        {
           String line = nameInputStream.nextLine( );
           String[] fields = line.split("\t");
           Integer AM = Integer.parseInt(fields[0]);
           String name = fields[1];
           namesHash.put(AM,name);
        }

        nameInputStream.close( );

        while (gradesInputStream.hasNextLine( ))
        {
           String line = gradesInputStream.nextLine( );
           String[] fields = line.split("\t");
           Integer AM = Integer.parseInt(fields[0]);
           String grade = fields[1];
           if (!nameHash.containsKey(AM)){ continue;}
           String name = namesHash.get(AM);
           outputStream.println(AM+"\t"+name+"\t"+grade);
        }

        gradesInputStream.close();
        outputStream.close( );
    }
}
```

Διάβασε τα ζεύγη ΑΜ, όνομα και βάλε τα σε ένα HashMap με κλειδί το ΑΜ

Υποθέτουμε ότι το κάθε ΑΜ εμφανίζεται μόνο μία φορά

Διάβασε τα ζεύγη ΑΜ, βαθμός και έλεγξε αν το ΑΜ εμφανίζεται ως κλειδί στο HashMap.

Αν ναι τύπωσε ΑΜ, όνομα και βαθμό στο αρχείο εξόδου

ΣΤΑΤΙΚΕΣ ΜΕΘΟΔΟΙ

Στατικές μέθοδοι

- Τι σημαίνει το keyword static στον ορισμό της main μεθόδου? Τι είναι μια στατική μέθοδος?
- Μια στατική μέθοδος μπορεί να κληθεί χωρίς αντικείμενο της κλάσης, χρησιμοποιώντας κατευθείαν το όνομα της κλάσης
  - Η μέθοδος ανήκει στην κλάση και όχι σε κάποιο συγκεκριμένο αντικείμενο.
  - Όταν καλούμε την συνάρτηση main κατά την εκτέλεση του προγράμματος δεν δημιουργούμε κάποιο αντικείμενο της κλάσης
  - Χρήσιμο για τον ορισμό βοηθητικών μεθόδων

Συντακτικό

- Ορισμός

```
class myClass
{
        ...

        public static ReturnType methodName(arguments)
        { … }

        ...
}
```

- Κλήση

```
myClass.methodName(arguments)
```

Παράδειγμα

Ορισμός

```
class Auxiliary
{
   public static int max(int x, int y){
       if (x > y){
           return x;
       }
       return y;
   }
}
```

Κλήση

```
int m = Auxiliary.max(6,5);
```

Η κλήση της μεθόδου max δεν χρειάζεται τον ορισμό αντικείμενου
Γίνεται χρησιμοποιώντας κατευθείαν το όνομα της κλάσης

Παρένθεση

- Ένας άλλος τρόπος να υλοποιήσετε το max τελεστή

```
public static int max(int x, int y){
      return (x>y)? x: y;
}
```

Η έκφραση:

```
condition ? value_if_true: value_if_false
```

επιστέφει μια τιμή ανάλογα με την αποτίμηση του condition και είναι ένας γρήγορος τρόπος να υλοποιήσουμε ένα if το οποίο επιστρέφει μία τιμή

Στατικές μεταβλητές

- Παρόμοια με τις στατικές μεθόδους μπορούμε να ορίσουμε και στατικές μεταβλητές
  - Οι στατικές μεταβλητές ανήκουν στην κλάση και όχι σε κάποιο συγκεκριμένο αντικείμενο και, εφόσον είναι public μπορούμε να έχουμε πρόσβαση σε αυτές χρησιμοποιώντας το όνομα της κλάσης χωρίς να έχουμε ορίσει κάποιο αντικείμενο.

Συντακτικό

- Ορισμός

```
class myClass
{
        public static Type varName;

        public static ReturnType methodName(arguments)
        { … }

        ...
}
```

- Κλήση

```
…. myClass.varName…. ;
```

Παράδειγμα

Ορισμός

```
class Auxiliary
{
    public static int factor = 2.0;

    public static int max(int x, int y){
        if (x > y){
            return x;
        }
        return y;
    }
}
```

Κλήση

```
int m =
    Auxiliary.factor * Auxiliary.max(6,5);
```

Σταθερές

- Οι στατικές μεταβλητές πολλές φορές χρησιμοποιούνται για να ορίσουμε σταθερές.
  - Τις ορίζουμε σε μία κλάση και μπορούμε να τις χρησιμοποιούμε σε διάφορα σημεία στο πρόγραμμα.

- Για να προσδιορίσουμε ότι μία μεταβλητή είναι σταθερά μπορούμε να χρησιμοποιήσουμε το keyword final.

Παράδειγμα

Ορισμός

```
class Circle
{
    public static final double PI = 3.14;

    public static double area(double r){
        return PI*r*r;
    }
}
```

Κλήση

```
int unitCircleArea = Circle.area(1);
System.out.println("PI value is" + Circle.PI);
```

Στατικές μέθοδοι

- Όταν ορίζουμε μια στατική μέθοδο μέσα σε μία κλάση, δεν μπορούμε να χρησιμοποιούμε μη στατικά πεδία, ή να καλούμε μη στατικές μεθόδους.
  - Μη στατικά πεδία και μη στατικές μέθοδοι συσχετίζονται με ένα αντικείμενο. Εφόσον μπορούμε να καλέσουμε μια στατική μέθοδο χωρίς αντικείμενο, δεν μπορούμε μέσα σε αυτή να χρησιμοποιούμε μη στατικά πεδία ή μεθόδους.
  - Σκεφτείτε ότι για κάθε χρήση μιας μεθόδου ή μιας μεταβλητής μπορούμε να βάλουμε το this μπροστά. Αν δεν υπάρχει αντικείμενο η αναφορά this δεν ορίζεται
- Αν θέλουμε να καλέσουμε μια μη στατική μέθοδο θα πρέπει να ορίσουμε ένα αντικείμενο μέσα στην στατική μέθοδο

Παράδειγμα

```
class Auxiliary2
{
        private int x;
        private int y;

        public Auxiliary2(int x, int y){
                this.x = x;
                this.y = y;
        }

        public int max(){
                return (x>y)? x: y;
        }

        public int min(){
                return (x>y)? y: x;
        }

        public static double maxToMin(int x, int y){
                Auxiliary2 aux = new Auxiliary2(x,y);
                return ((double)aux.max())/aux.min();
        }
}
```

Στατικές μεταβλητές

- Εκτός από σταθερές μπορούμε να ορίσουμε στατικές μεταβλητές όταν θέλουμε διαφορετικά αντικείμενα να επικοινωνούν μέσω μιας μεταβλητής
  - Υπάρχει μόνο ένα αντίγραφο μιας στατικής μεταβλητής, άρα όταν το αλλάζει ένα αντικείμενο την αλλαγή την βλέπουν και όλα τα άλλα αντικείμενα της κλάσης.
- Παράδειγμα: Στο πρόγραμμα TakeTurns δείχνουμε πως μπορούμε να χρησιμοποιήσουμε στατικές μεταβλητές για να επικοινωνούν μεταξύ τους τα αντικείμενα.

```
class TakeTurns
{
    private static int players = 0;
    private static int rounds = 0;
    private int id;

    public TakeTurns(int i){
        id = i;
        players ++;
    }

    public void play(){
        if (rounds%players == id){
            System.out.println("Round "+ rounds + " Player " + id + " played");
            rounds ++;
        }
    }

    public static void main(String args[]){
        TakeTurns player0 = new TakeTurns(0);
        TakeTurns player1 = new TakeTurns(1);

        for (int i = 0; i < 10; i ++){
            player0.play();
            player1.play();
        }
    }
}
```

Τα αντικείμενα player0 και player1 βλέπουν τις ίδιες μεταβλητές players και rounds, αλλά διαφορετική μεταβλητή id

Ο κάθε παίχτης παίζει μόνο όταν είναι η σειρά του

Στατικές μέθοδοι και μεταβλητές

- Έχετε ήδη χρησιμοποιήσει στατικές μεθόδους και μεταβλητές σε διάφορες περιπτώσεις

- Παραδείγματα
  - System.out: στατικό πεδίο της κλάσης System, το οποίο κρατάει ένα PrintStream με το οποίο μπορούμε γράψουμε στην οθόνη.
  - System.in: στατικό πεδίο της κλάσης System, το οποίο κρατάει ένα FileInputStream που συνδέεται με το πληκτρολόγιο.
  - System.exit(): στατική μέθοδος της κλάσης System

Περιβάλλουσες κλάσεις

- Οι wrapper classes Integer, Double, Boolean και Character έχουν πολλές στατικές μεθόδους και στατικά πεδία που μας βοηθάνε να χειριζόμαστε τους βασικούς τύπους.
  - Integer.parseInt(String): Μετατρέπει ένα String σε int.
    - Αντίστοιχα: Double.parseDouble(String), Boolean.parseBoolean(String)
  - Integer.MAX_VALUE, Integer.MIN_VALUE: Μέγιστη και ελάχιστη τιμή ενός ακεραίου
    - Αντίστοιχα: Double.MAX_VALUE, Double.MIN_VALUE
  - Character.isDigit(char): επιστρέφει true αν ο χαρακτήρας είναι ένα ψηφίο
    - Παρόμοια: Character.isLetter(char), Character.isLetterOrDigit(), Character.isWhiteSpace(char)
- Οι κλάσεις αυτές έχουν και μη στατικές μεθόδους.

Η κλάση Math

- Μία κλάση με πολλές στατικές μεθόδους και στατικά πεδία για μαθηματικούς υπολογισμούς
- Παραδείγματα
  - min: επιστρέφει το ελάχιστο δύο αριθμών
  - max: επιστρέφει το μέγιστο δύο αριθμών
  - abs: επιστρέφει την απόλυτη τιμή
  - pow(x,y): υψώνει το x στην y δυναμη
  - floor/ceil: επιστρέφει τον μεγαλύτερο/μικρότερο ακέραιο που είναι μικρότερος/μεγαλυτερος από το όρισμα
  - sqrt: επιστρέφει την τετραγωνική ρίζα ενός αριθμού
  - PI: ο αριθμός π
  - E: Η βάση των φυσικών λογαρίθμων

Συμπερασματικά

- Στατικές μεθόδους και πεδία συνήθως ορίζουμε όταν θέλουμε μια βοηθητική συλλογή από σταθερές και μεθόδους (παρόμοια με την κλάση Math της Java).
- Μια στατική μέθοδο που μπορείτε να ορίσετε για κάθε κλάση είναι η main, ώστε να τεστάρετε μια συγκεκριμένη κλάση.

ΕΣΩΤΕΡΙΚΕΣ ΚΛΑΣΕΙΣ

Εσωτερικές κλάσεις

- Μπορούμε να ορίσουμε μια κλάση μέσα στον ορισμό μιας άλλης κλάσης

```
class Shape
{
        private class Point
        {

                <Code for Point>
        }

        <Code for Shape>
}
```

Γιατί να το κάνουμε αυτό?

- Η κλάση Point μπορεί να είναι χρήσιμη μόνο για την Shape
- Μας επιτρέπει να ορίσουμε άλλη Point σε άλλο σημείο

- Η Point και η Shape έχουν η μία πρόσβαση στα ιδιωτικά πεδία και μεθόδους της άλλης