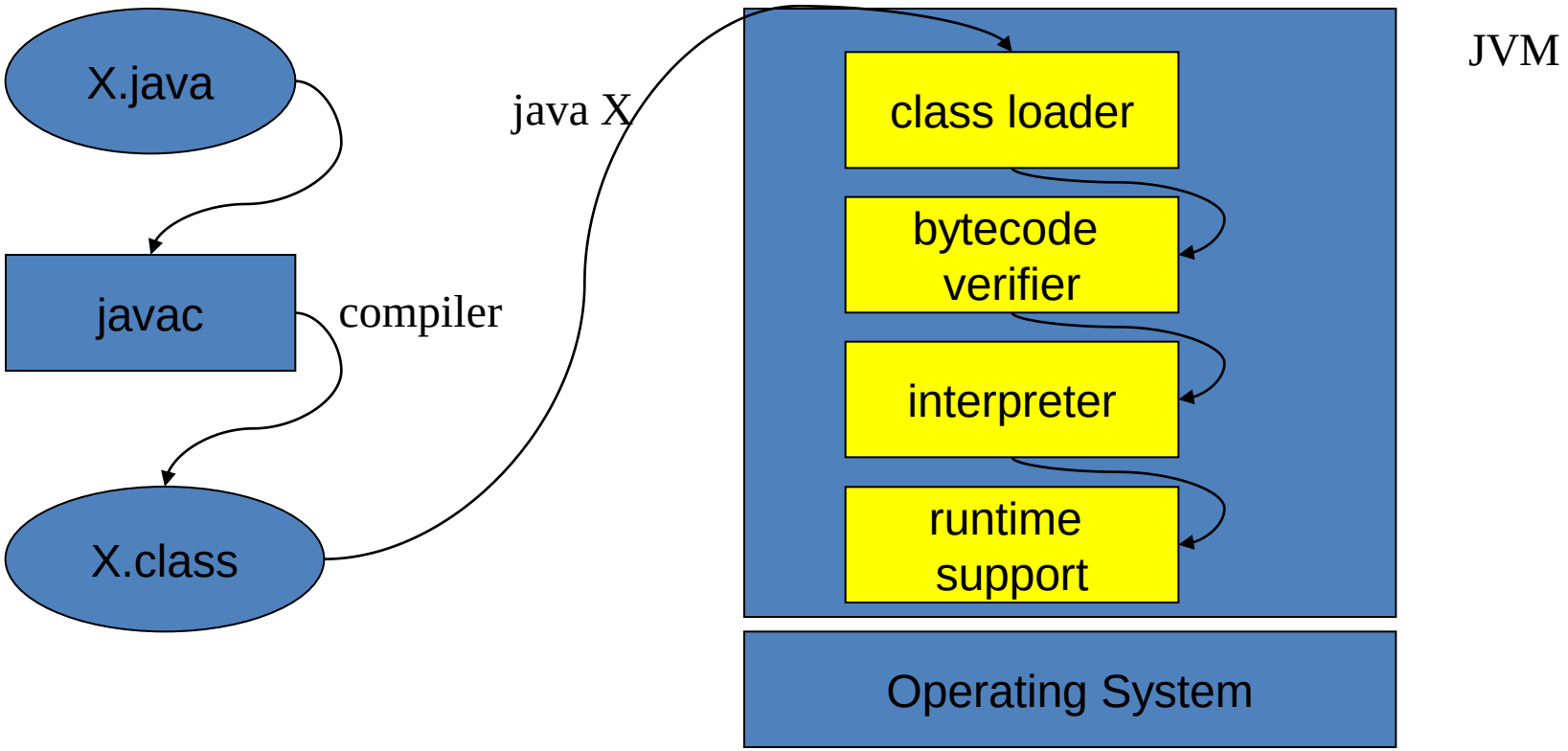
JAVA, NETBEANS

ΑΝΑΚΕΦΑΛΑΙΩΣΗ

Τα βασικά για την γλώσσα Java

Java portability

- Το μεγαλύτερο πλεονέκτημα της Java είναι η μεταφερσιμότητα: ο κώδικας μπορεί να τρέξει πάνω σε οποιαδήποτε πλατφόρμα.
  - Write-Once-Run-Anywhere μοντέλο, σε αντίθεση με το σύνηθες Write-Once-Compile-Anywhere μοντέλο.
- Αυτό επιτυγχάνεται δημιουργώντας ένα ενδιάμεσο κώδικα (bytecode) ο οποίος μετά τρέχει πάνω σε μια εικονική μηχανή (Java Virtual Machine) η οποία το μεταφράζει σε γλώσσα μηχανής.
  - Οι προγραμματιστές πλέον γράφουν κώδικα για την εικονική μηχανή, η οποία δημιουργείται για οποιαδήποτε πλατφόρμα.


X.java
javac
compiler
X.class
java X
class loader
bytecode verifier
interpreter
runtime support
Operating System
JVM

Δομή προγράμματος

- Γενικά ένα απλό πρόγραμμα υλοποιείται σε ένα αρχείο X.java αποτελείται από
  - Ένα βασικό δομικό στοιχείο – κλάση - που ορίζεται ως public class X και περιλαμβάνει τον κώδικα της main του προγράμματος.
    - όνομα κλάσης == όνομα αρχειου !!…
    - πρέπει η main να δηλωθεί ως static void.
    - δέχεται σαν παραμέτρους από τη γραμμή εντολών ένα πίνακα από String.
    - το αρχείο μπορεί να περιέχει και άλλες μεθόδους
  - άλλα επιπλέον δομικά στοιχεία – κλάσεις - που ορίζονται ως non public κλάσεις που υλοποιούν τον υπόλοιπο κώδικα του προγράμματος.
- Τα πάντα υλοποιούνται μέσα σε κλάσεις.

Compilation and Execution

- Compilation
  - javac TestPerson.java
  - Παράγει το μεταγλωττισμένο αρχείο TestPerson.class
- Execution
  - java TestPerson 70 170 80 160
  - Εκτελεί το πρόγραμμα με παραμέτρους εισόδου 70 170 80 160

Κοινές μέθοδοι

- Όλες οι κλάσεις στην Java έχουν κοινό πρόγονο την κλάση Object, η οποία είναι στην κορυφή της ιεραρχίας της Java.
- Η κλάση Object έχει κάποιες μεθόδους τις οποίες κληρονομούν όλες οι κλάσεις. Δύο από αυτές που είναι χρήσιμες:
  - toString(): μετατρέπει ένα αντικείμενο σε αλφαριθμητικό. Δεν έχει πάντα νόημα, αλλά σε ορισμένες περιπτώσεις είναι πολύ βολικό.
  - equals(Object): ελέγχει για ισότητα μεταξύ δύο αντικειμένων του ίδιου τύπου.

Ορισμός κλάσεων

- Ο ορισμός των μεθόδων και των πεδίων γίνεται μαζί με τη δήλωση.

```
class Person {
 private int weight = 4;
 private int height = 50;

  public Person(int w, int h) { // Constructor
        weight = w;
        height = h;
   }

   public void statistics(){ // Computes Ratio
        System.out.println(
            "Statistics for person : " + height/weight);
   }
}
```

Δεν χρειάζεται “;”

Μεταβλητές

- Εκτός από τις μεταβλητές βασικού τύπου δεδομένων, όλες οι υπόλοιπες μεταβλητές είναι
  - Αναφορές σε αντικείμενα
    - κλάσης που προσφέρεται από τη Java (πχ String)
    - κλάσης του προγράμματος που κατασκευάζουμε εμείς
  - Διεύθυνση πίνακα
- Όλες οι μεταβλητές αρχικοποιούνται αυτόματα εκτός και αν τις αρχικοποιήσουμε κατά τη δήλωση
  - Οι μεταβλητές βασικού τύπου σε μηδέν.
  - Οι μεταβλητές μη βασικού τύπου σε null.
- Αρχικοποιούνται σε οποιοδήποτε σημείο του κώδικα,
  - είτε με χρήση της new που δεσμεύει χώρο για ένα νέο αντικείμενο, πίνακα, string
  - είτε με εκχώρηση της διεύθυνσης που περιέχεται σε μια άλλη μεταβλητή του ίδιου τύπου.
- Εκτός από τις μεταβλητές βασικού τύπου οι υπόλοιπες μεταβλητές δεσμεύουν χώρο από το heap και όχι το stack της μνήμης.

Wrapper Classes

- Η Java ορίζει wrapper classes για τους βασικούς τύπους δεδομένων.
  - Integer για τον int
  - Float για το float
  - κλπ
- Οι κλάσεις αυτές μας δίνουν επιπλέον ευελιξία στον χειρισμό των βασικών τύπων.

```
public String ratio(){ // makes string out of Ratio
    Integer w = weight;
    Integer h = height;
    String s = h.toString() + "/" + w.toString();
    return s;
 }
```

Constructors

- Όπως και στην C++ αν δεν προσδιορίσουμε τον Constructor, τότε χρησιμοποιείται ο default.
- Στην περίπτωση της Java, ο default constructor κάνει κάτι: αρχικοποιεί όλα τα πεδία σε «μηδενική» τιμή.
- Όπως και στην C++ αν ορίσουμε ένα constructor, τότε ο default παύει να υφίσταται.

Destructors?

- Στην Java δεν χρειαζόμαστε destructors!
- Η αποδέσμευση της μνήμης γίνεται αυτόματα από το garbage collector ο οποίος φροντίζει για τη διαγραφή των αντικειμένων που δεν χρησιμοποιούνται πλέον.
- Αν ο χρήστης θέλει να δώσει μια ένδειξη στον garbage collector μπορεί να χρησιμοποιήσει την εντολή finalize().

Παράδειγμα

- Θα κατασκευάσουμε σε java το παράδειγμα από την διάλεξη 2:
  - Δύο οχήματα κινούνται πάνω σε μία ευθεία.
  - Σε κάθε βήμα διαλέγουν τυχαία, αν θα πάνε αριστερά, δεξιά, ή θα μείνουν στην ίδια θέση.
  - Η προσομοίωση τελειώνει όταν συγκρουστούν.

```
import java.lang.* ;
import java.util.* ;

class Car
{
        private int pos;
        public Car()
        {
                pos = 0;
        }
        public Car(int x)
        {
                pos = x;
        }
        public void move()
        {
                Random r = new Random();
                pos += r.nextInt(2) - 1;
        }
        public int GetPos()
        {
                return pos;
        }
}
```

```
public class CarGame{

        private static Car x = new Car();
        private static Car y = new Car();

        public static void main(String[] args)
        {
                int iter = 1;
                x.move();
                y.move();
                while (x.GetPos() != y.GetPos()){
                        x.move();
                        y.move();
                        iter ++;
                }
                System.out.println("Colision after "
                        + iter + "moves at position "
                        + x.GetPos());
        }
}
```

Η δήλωση της main πρέπει να είναι ακριβώς έτσι

Αρχείο CarGame.java

```
import java.lang.* ;
import java.util.* ;

class Car
{
  private int pos;

  public Car(){
    pos = 0;
  }
  public Car(int x){
    pos = x;
  }
  public void move(){
    Random r = new Random();
    pos += r.nextInt(2) - 1;
  }
  public int GetPos(){
    return pos;
  }
}

public class CarGame{

      private static Car x = new Car();
      private static Car y = new Car();

      public static void main(String[] args){
           int iter = 1;
           x.move(); y.move();
           while (x.GetPos() != y.GetPos()){
             x.move(); y.move();
             iter ++;
           }
           System.out.println("Colision after " + iter + "moves at position " + x.GetPos());
      }
}
```

NETBEANS

IDEs

- IDE: Integrated Development Environment
  - Ένα πρόγραμμα που για τη δημιουργία λογισμικού. Παρέχει τις εξής λειτουργίες:
    - Code Editor
    - Compliler/Interpreter
    - Build Enviroment
    - Debugger
  - Συνήθως, τα IDEs χρειάζονται κάποιο χρόνο για να τα μάθεις αλλά όταν εξοικειωθείς με αυτά ο προγραμματισμός γίνεται πολύ πιο εύκολος (και το debugging ακόμη περισσότερο).
    - Μειονέκτημα: Είναι λίγο φασαρία για την δημιουργία απλών μικρών προγραμμάτων.
- Γνωστά IDEs:
  - Microsoft Visual Studio, Eclipse, Netbeans

Netbeans

- Ο κώδικας είναι οργανωμένος σε projects, στα οποία προσθέτουμε κλάσεις, ή βιβλιοθήκες.
- Για κάθε project δημιουργείται ένα folder το οποίο περιέχει τα αρχεία του project και το αποτέλεσμα του compilation
  - src folder: περιέχει τα .java αρχεία με τον πηγαίο κώδικα.
  - build folder: περιέχει τα .class αρχεία με τον ενδιάμεσο κώδικα.
  - dist folder: περιέχει ένα .jar αρχείο το οποίο μπορουμε να κάνουμε distribute.

Λειτουργίες

- Build: Κάνει compile το project.
- Run: Τρέχει τον κώδικα
  - Για τον ορισμό των παραμέτρων του προγράμματος (line arguments):
    - Δεξί κλικ στο project -> Properties -> Run -> Arguments
- Debug: Μας επιτρέπει να τρέξουμε το πρόγραμμα γραμμή γραμμή για να βρούμε λάθη.
- Refactor: Επιτρέπει εύκολα αλλαγές στον κώδικα
  - Π.χ. αλλαγή ονόματος μεταβλητών.
- Profile: Μας βοηθάει να δούμε που καταναλώνονται τα resources του προγράμματος

ΕΠΙΠΛΕΟΝ ΣΤΟΙΧΕΙΑ ΤΗΣ ΓΛΩΣΣΑΣ JAVA

Interfaces, Containers

Εξαιρέσεις

- Ένα exception είναι ένα αντικείμενο της κλάσης Exception (ή παραγόμενων κλάσεων).
- Το αντικείμενο δημιουργείται και επιστρέφεται με την εντολή throw (αντί της return) από μία μέθοδο όταν προκύπτει μια μη φυσιολογική κατάσταση η οποία επιβάλει να διακοπεί η εκτέλεση αυτής της μεθόδου. Π.χ.
  - Διαίρεση με το μηδέν
  - Πρόσβαση μίας null αναφοράς
  - Λάθος σε I/O
- Εκτός από τις εξαιρέσεις που «πετάνε» οι μέθοδοι της Java, ο χρήστης μπορεί πετάει και δικά του exceptions.

Χειρισμός εξαιρέσεων

- Το **`try-catch`** block

```
try{
        // … Κώδικας που μπορεί να πετάξει καποια εξαίρεση
}
catch (Exception e){
        System.out(“caught the exception”);
}
```

- Μπορούμε να έχουμε ξεχωριστά catch για διαφορετικούς τύπους εξαιρέσεων.
  - IOException
  - NullPointerException
  - IllegalArgumentException
  - RunTimeException

Χειρισμός εξαιρέσεων

- Μία εξαίρεση μπορούμε να την χειριστούμε επί τόπου, είτε να την κάνουμε rethrow στη μέθοδο που κάλεσε τη δική μας μέθοδο.
- Δεν είναι απαραίτητο να χειριστούμε όλες τις πιθανές εξαιρέσεις (αν και πρέπει γιατί αλλιώς το πρόγραμμα μας δεν θα είναι ασφαλές), αλλά υπάρχουν κάποιες για τις οποίες η Java απαιτεί να τις χειριστούμε (αλλιώς ο compiler χτυπάει λάθος)
  - Τα IOExceptions είναι ένα παράδειγμα εξαιρέσεων που πρέπει να χειριστουμε οπωσδηποτε.

IO – Read/Write lines from/to Standard Input/Output

Υποχρωτικά μέσα σε try-catch block

```
import java.lang.*;
import java.io.*;

public class IOReadlines {

 public static void main (String args[]){
  try{
      BufferedReader br = new BufferedReader(
                              new InputStreamReader(System.in));
      String line;

      while ((line = br.readLine()) != null){
          System.out.println(line);
      }
  } catch (IOException e) {
        System.out.println("Input error");
        System.exit(1);
  }
   }
}
```

IO – Read/Write lines from/to Files

```
import java.io.*;

public class Main {
    public static void main(String[] args) {
        try{
            BufferedReader inReader = new BufferedReader(
                                        new FileReader("Files/in.txt"));
            PrintWriter outWriter = new PrintWriter(
                                        new FileWriter("Files/out.txt"));

            String line;
            while((String line = inReader.readLine())!=null){
                outWriter.println(line);
                System.out.println(line);
            }
            outWriter.close();
        } catch(IOException ex){
            System.out.println("IO Error" + ex);
        }
    }
}
```

Κληρονομικότητα και πολυμορφισμός

- Η κληρονομικότητα ορίζεται με το keyword extends
  - `class Employee extends Person { … }`
- Πολυμορφισμός: το late binding είναι η default συμπεριφορά του JVM στην εκτέλεση των προγραμμάτων.
  - Η κλάση του αντικειμένου εξαρτάται από την κλάση του αντικειμένου που χρησιμοποιούμε και όχι από την κλάση του αντικειμένου στον ορισμό

Παράδειγμα

```
import java.lang.*;
import java.io.*;

class Person{
    private  String fname;
    private String lname;

    public Person(String fn, String ln){
      fname = fn;
      lname = ln;
     }

   public String getPersonalDetails(){
      return fname + lname;
     }
}
```

Παράδειγμα

```
class Employee extends Person {
    private int basicSalary;

    public Employee( String fn, String ln, int sal){
      super(fn, ln);
      basicSalary = sal;
    }

    public int getSalary(){
      return basicSalary;
    }
}
```

Abstract Classes

- Παρόμοια με τις virtual classes στην C++, αν ορίσω μία μέθοδο ως abstract, τότε όλη η κλάση γίνεται abstract και πλέον δεν μπορώ να ορίσω αντικείμενα αυτής της κλάσης.

Παράδειγμα

```
import java.lang.*;
import java.io.*;

abstract class Person{
    private  String fname;
    private String lname;

    public Person(String fn, String ln){
      fname = fn;
      lname = ln;
     }

   public abstract String getPersonalDetails();
    }
}
```

Παράδειγμα

```
class Employee extends Person {
    private int basicSalary;

    public Employee( String fn, String ln, int sal){
      super(fn, ln);
      basicSalary = sal;
    }

    public int getSalary(){
      return basicSalary;
    }

    public String getPersonalDetails(){
      return fname + " "+ lname + " " + basicSalary.toString();
    }

}
```

Interfaces

- Τα Interfaces πηγαίνουν την ιδέα της αφηρημένης κλάσης ένα βήμα παραπέρα.
- Σε ένα Interface έχουμε μόνο:
  - Ορισμούς μεθόδων χωρίς την υλοποίηση.
    - Οι μέθοδοι είναι by default public
  - Ορισμούς μεταβλητών οι οποίες είναι by default static.
- Συντακτικό:

```
interface INTERFACE_NAME
{
        … methods …
}
```

Παράδειγμα

Η κλάση Employee implements το interface Person

```
import java.lang.*;
import java.io.*;

Interface Person{
     String getPersonalDetails();
}

class Employee implements Person {
    private int basicSalary;
    private String fname, lname;

    public Employee( String fn, String ln, int sal){
      fname = fn;
      lname = ln;
      basicSalary = sal;
    }

    public String getPersonalDetails(){
      return fname + " "+ lname + " " + basicSalary.toString();
    }

}
```

Παράδειγμα

```
class Employee extends Person {
    private int basicSalary;

    public Employee( String fn, String ln, int sal){
      super(fn, ln);
      basicSalary = sal;
    }

    public int getSalary(){
      return basicSalary;
    }

    public String getPersonalDetails(){
      return fname + " "+ lname + " " + basicSalary.toString();
    }

}
```

Containers

- Παρόμοια με την C++ η Java έχει μια βιβλιοθήκη από Containers την οποία μπορούμε να χρησιμοποιήσουμε για να αποθηκεύουμε και να επεξεργαζόμαστε δεδομένα.
  - Array
  - ArrayList
  - LinkedList
  - Set
  - Map
  - PriorityQueue

ArrayList

- Ο Container ArrayList κληρονομει από το List και αυτό από το Collection.
- Προσφέρει σειριακή αποθήκευση δεδομένων και έχει όλα τα πλεονεκτήματα και μειονεκτήματα του vector στην C++.
- Στην Java δεν επιτρέπεται υπερφόρτωση τελεστών οπότε χρησιμοποιούμε την μέθοδο get(index) για να διαβάσουμε ένα στοιχείο.
- Διάσχιση του ArrayList με την foreach εντολή είναι πιο απλή απ ότι με τον iterator.

```
import java.io.*;
import java.util.*;

public class arraylist {
    public static void main(String[] args) {
       ArrayList<Integer> A = new ArrayList<Integer>();
       for (int i = 0; i < 10; i ++){
              Random r = new Random();
              A.add(r.nextInt(100));
              System.out.println(A.get(i));
       }
       Collections.sort(A);
       System.out.println("");
       for (int x: A){
              System.out.println(x);
       }
       System.out.println("");
       System.out.println(A.toString());
    }
}
```

HashMap

- Το HashMap ορίζει ένα συνειρμικό αποθηκευτή (associative container) ο οποίος συσχετίζει κλειδιά με τιμές, κληρονομεί από την πιο γενική κλάση Map.
  - Π.χ., ο βαθμός ενός φοιτητή, η συχνότητα με την οποία εμφανίζεται μια λέξη σε ένα κείμενο.
- Η βιβλιοθήκη της Java μας δίνει πιο εύκολη πρόσβαση στα κλειδιά και τις τιμές του map.
- Χρήσιμες μέθοδοι:
  - put(key,value): προσθέτει ένα νέο key-value ζεύγος
  - containsKey(key): επιστρέφει αληθές αν υπάρχει το κλειδί
  - containsValue(value): επιστρέφει αληθές αν υπάρχει η τιμή
  - values(): επιστρέφει ένα Collection με τις τιμές
  - keySet(): επιστρέφει ένα Set με τις τιμές.

```
import java.io.*;
import java.util.*;

public class mapexample {
    public static void main(String[] args) {
        String line;
        Map<String,Integer> namesGrades = new HashMap<String,Integer>();
        try{
            FileReader fr = new FileReader("Files/in.txt");
            BufferedReader inReader = new BufferedReader(fr);

            while((line = inReader.readLine())!=null){
                System.out.println(line);
                String [] words = line.split("\t");
                Integer grade = Integer.parseInt(words[1]);
                namesGrades.put(words[0],grade);
            }
        } catch(IOException ex){
            System.out.println("IO Error" + ex);
        }
        for(String x: namesGrades.keySet()){
                System.out.println(x + " -- " + namesGrades.get(x));
        }
    }
}
```