



ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ
ΑΝΑΤΟΛΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ ΚΑΙ ΘΡΑΚΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΤΕ

Προγραμματισμός Διεπαφής Χρήστη

Ενότητα 1: Ανασκόπηση της Java

Κωνσταντίνος Τσίκνας
ktsik@teiemt.gr

Περιεχόμενα ενότητας














- Εργαλεία ανάπτυξης εφαρμογών σε Java
 - Προαπαιτούμενα προγράμματα
 - Βασικά στοιχεία της JAVA
 - Δημιουργία πρώτου προγράμματος JAVA
 - Εισαγωγή στο Netbeans IDE
- Αντικειμενοστραφής προγραμματισμός σε Java
 - Η έννοια της κλάσης και του αντικειμένου
 - Δομητές (Constructors)
 - Υπερφόρτωση μεθόδων (overloading)
 - Κληρονομικότητα (Inheritance)
 - Πολυμορφισμός (polymorphism)
 - Αφηρημένες κλάσεις (abstract classes)

Περιεχόμενα ενότητας

- Εργαλεία ανάπτυξης εφαρμογών σε Java
 - Προαπαιτούμενα προγράμματα
 - Βασικά στοιχεία της JAVA
 - Δημιουργία πρώτου προγράμματος JAVA
 - Εισαγωγή στο Netbeans IDE
- Αντικειμενοστραφής προγραμματισμός σε Java
 - Η έννοια της κλάσης και του αντικειμένου
 - Δομητές (Constructors)
 - Υπερφόρτωση μεθόδων (overloading)
 - Κληρονομικότητα (Inheritance)
 - Πολυμορφισμός (polymorphism)
 - Αφηρημένες κλάσεις (abstract classes)

Απαιτούμενα προγράμματα (1/2)

- Java Platform Standard Edition, διαθέσιμη στην σελίδα <https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
- Επιλέξτε την έκδοση που είναι κατάλληλη στο λειτουργικό σύστημα του υπολογιστή σας.

Java SE Development Kit 8u191		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
<input type="radio"/> Accept License Agreement <input checked="" type="radio"/> Decline License Agreement		
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.97 MB	 jdk-8u191-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.92 MB	 jdk-8u191-linux-arm64-vfp-hflt.tar.gz
Linux x86	170.89 MB	 jdk-8u191-linux-i586.rpm
Linux x86	185.69 MB	 jdk-8u191-linux-i586.tar.gz
Linux x64	167.99 MB	 jdk-8u191-linux-x64.rpm
Linux x64	182.87 MB	 jdk-8u191-linux-x64.tar.gz
Mac OS X x64	245.92 MB	 jdk-8u191-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	133.04 MB	 jdk-8u191-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	94.28 MB	 jdk-8u191-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	134.04 MB	 jdk-8u191-solaris-x64.tar.Z
Solaris x64	92.13 MB	 jdk-8u191-solaris-x64.tar.gz
Windows x86	197.34 MB	 jdk-8u191-windows-i586.exe
Windows x64	207.22 MB	 jdk-8u191-windows-x64.exe

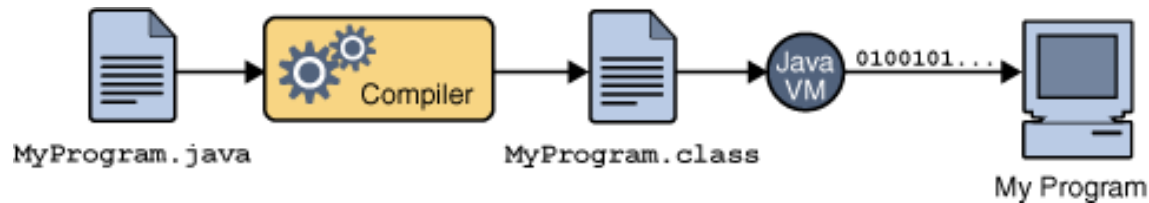
Απαιτούμενα προγράμματα (2/2)

- NetBeans IDE 8.0.2 <https://netbeans.org/downloads/8.0.2/>
- Επιλέξτε την έκδοση «Java SE» (ενσωματώνει την Java Fx)

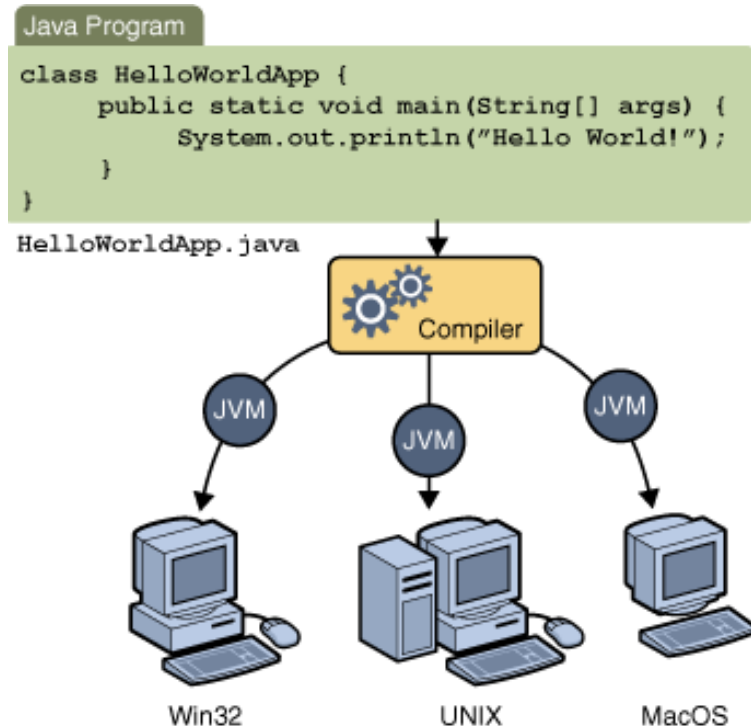
NetBeans IDE Download Bundles					
Supported technologies *	Java SE	Java EE	C/C++	HTML5 & PHP	All
NetBeans Platform SDK	•	•			•
Java SE	•	•			•
Java FX	•	•			•
Java EE		•			•
Java ME					•
HTML5		•		•	•
Java Card™ 3 Connected					•
C/C++			•		•
Groovy					•
PHP				•	•
Bundled servers					
GlassFish Server Open Source Edition 4.1		•			•
Apache Tomcat 8.0.15		•			•
	Download	Download	Download	Download	Download
	Free, 90 MB	Free, 186 MB	Free, 63 MB	Free, 63 MB	Free, 205 MB

Βασικά στοιχεία της Java (1/2)

- Διαδικασία Ανάπτυξης JAVA λογισμικού

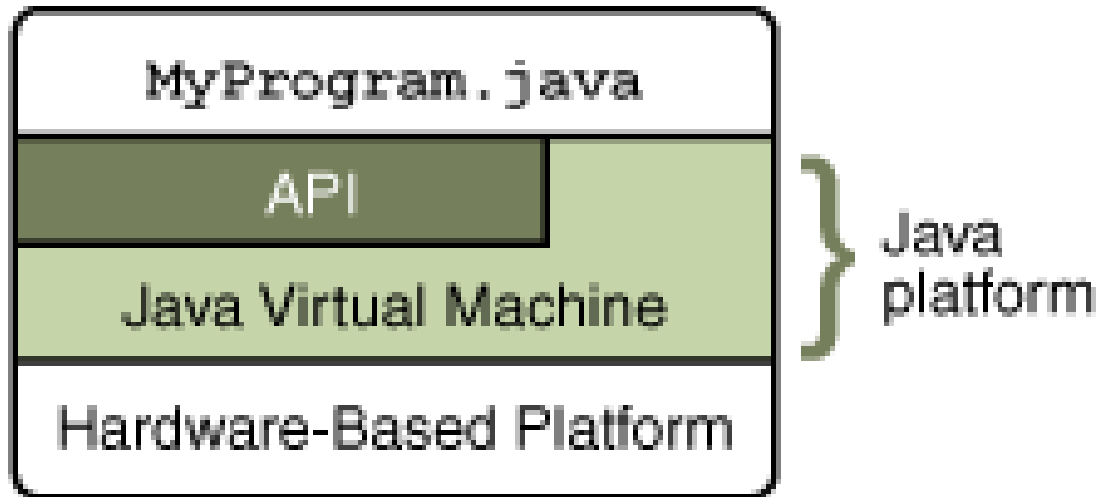


- Java Virtual Machine και ανεξαρτησία Πλατφόρμας



Βασικά στοιχεία της Java (2/2)

- JAVA Πλατφόρμα
 - *JAVA Application Programming Interface(API)*: Συλλογή κλάσεων που συμπεριλαμβάνονται στο περιβάλλον ανάπτυξης της Java
 - *Java Virtual Machine(JVM)*: Τρέχει το εκτελέσιμο πρόγραμμα Java Bytecode και το μεταφράζει σε γλώσσα μηχανής. Ανεξαρτησία από το hardware



Δημιουργία πρώτου προγράμματος (1/4)

- Δημιουργούμε ένα αρχείο HelloProgram.java και γράφουμε τον εξής κώδικα :

1

```
/*My first program*/
```

2

```
public class HelloProgram{
```

3

```
    public static void main(String[] str){
```

```
        System.out.println("Hello World!");
```

```
    }
```

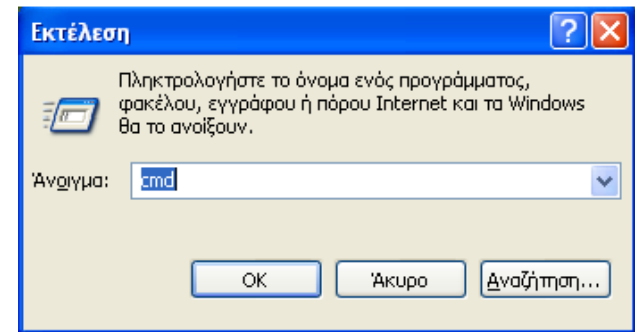
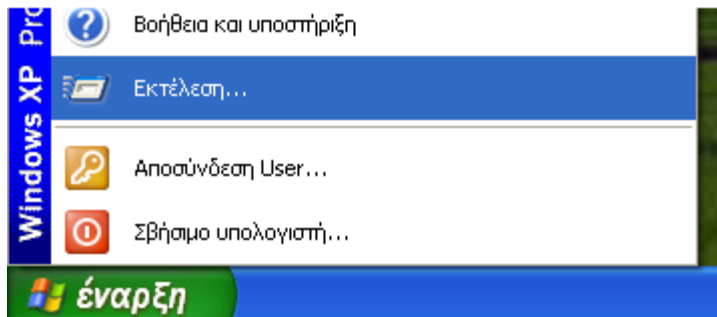
```
}
```

Προσοχή: Το όνομα του αρχείου και το όνομα της κλάσης. Πρέπει να είναι ακριβώς τα ίδια.

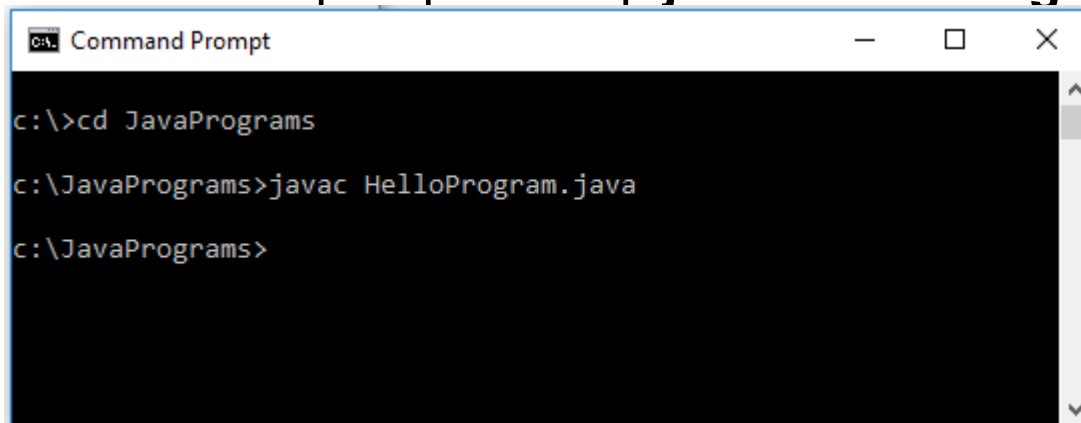
- Δημιουργούμε τη κλάση με όνομα HelloProgram. 1
- Δημιουργούμε την κύρια μέθοδο (main method). 2
- Γράφουμε τον κώδικα που θέλουμε να κάνει το πρόγραμμά μας. Στην συγκεκριμένη περίπτωση θέλουμε να εμφανίσει στο output την φράση «Hello World!». 3

Δημιουργία πρώτου προγράμματος (2/4)

- Δημιουργούμε ένα φάκελο με όνομα «javaprograms» στον C:\ και μεταφέρουμε το java αρχείο μας εκεί.
- Από το μενού πατάμε εκτέλεση (Run), όπου γράφουμε “cmd” για να μας ανοίξει ένα παράθυρο εντολών (command line):



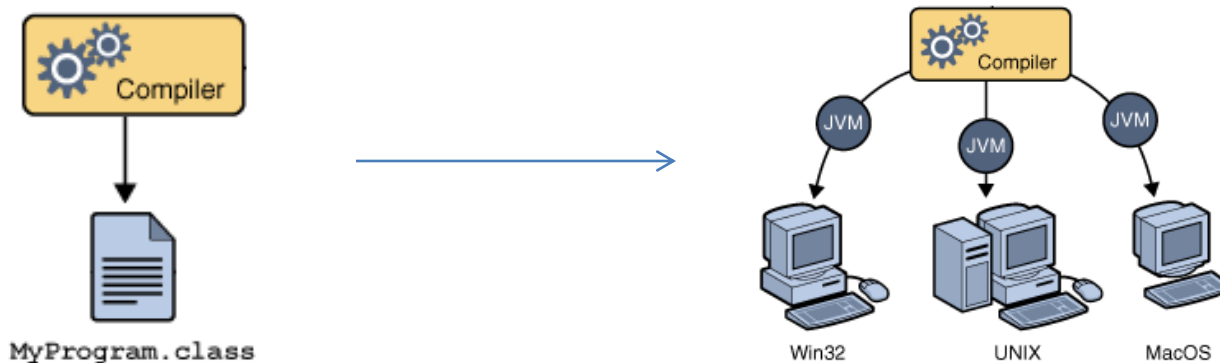
- Πάμε στη διαδρομή του δίσκου όπου αποθηκεύσαμε το πρόγραμμά μας και εκτελούμε την εντολή **javac HelloProgram.java**



Προσοχή: Αν μας εμφανίσει ότι «δεν αναγνωρίζεται ως εσωτερική ή εξωτερική εντολή» τότε ανατρέξτε στον οδηγό ([Set PATH](#))

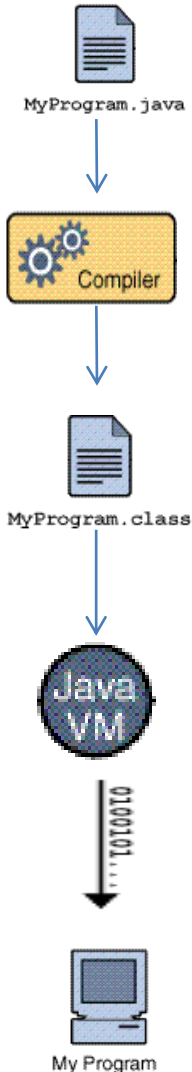
Δημιουργία πρώτου προγράμματος (3/5)

- Αν δεν μας εμφανίσει κάποιο σφάλμα σημαίνει ότι το πρόγραμμά μας έγινε compile επιτυχώς. Σε περίπτωση σφάλματος διορθώνουμε το λάθος και κάνουμε ξανά compile.
- Μόλις γίνει επιτυχώς compile το πρόγραμμά μας, δημιουργείται στον φάκελο που βρίσκεται το πρόγραμμα ένα αρχείο .class το οποίο έχει ίδιο όνομα με το πρόγραμμά μας και το οποίο περιέχει τον java κώδικά μεταγλωττισμένο σε κώδικα java μηχανής (JVM).



Δημιουργία πρώτου προγράμματος (5/5)

Στάδια δημιουργίας και εκτέλεσης προγράμματος



1. Δημιουργούμε το πρόγραμμα μας σε ένα αρχείο με τη κατάληξη .java (java αρχείο),

2. Κάνουμε Compile το πρόγραμμα.

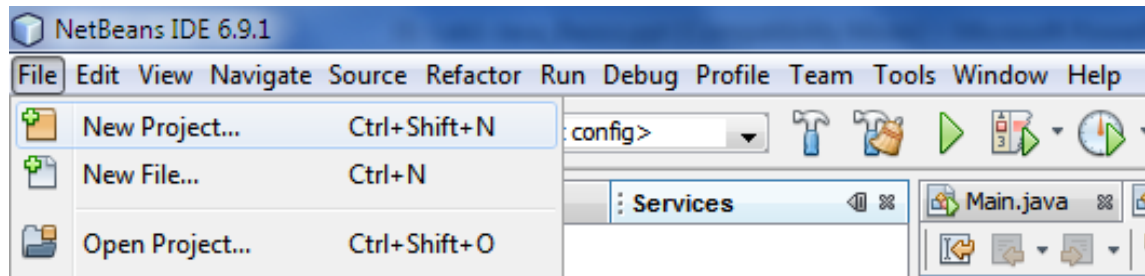
3. Δημιουργία αρχείου .class, το οποίο περιέχει τον java κώδικα σε γλώσσα του Virtual Machine (VM),

4. Το αρχείο class εκτελείται από το Java Virtual Machine,

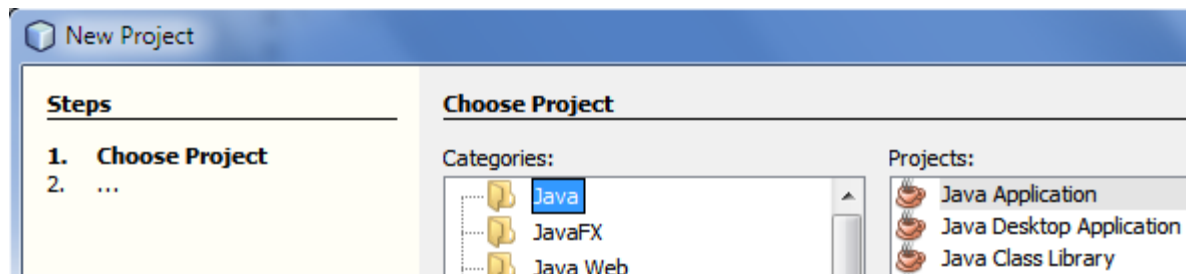
5. Το πρόγραμμά μας τρέχει στο λειτουργικό σύστημα του εκάστοτε υπολογιστή.

Εισαγωγή στο Netbeans (1/6)

- Δημιουργία ενός νέου project
 - Ανοίξτε το Netbeans IDE
 - Επιλέξτε File → New Project...



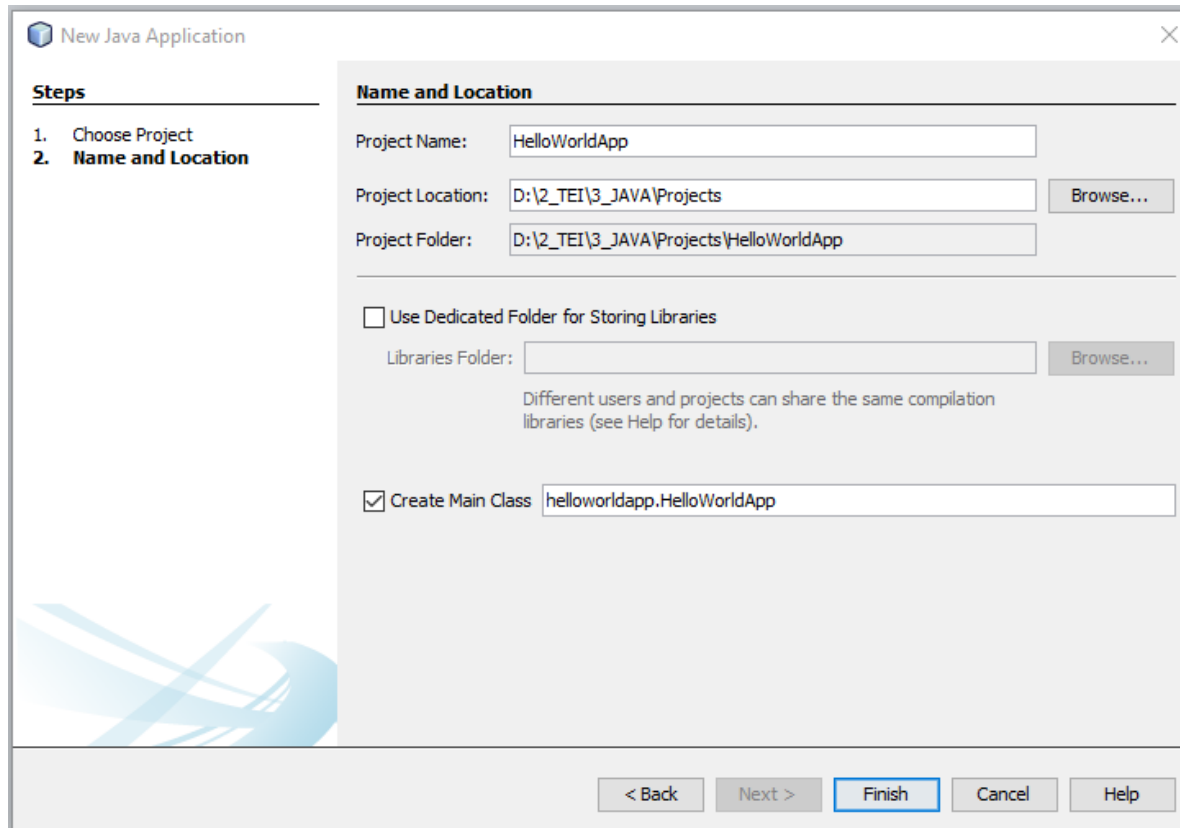
- Στον wizard που ανοίγει επιλέξτε την κατηγορία Java → Java Application και πατήστε next



Εισαγωγή στο Netbeans (2/6)

Στην σελίδα “Name and Location” του οδηγού, κάντε τα εξής

- Στο πεδίο “Project Name” δώστε **HelloWorldApp**
- Στο πεδίο “Project Location” επιλέξτε την τοποθεσία που θα σώζεται το project
- Αφήστε το κουτάκι “Create Application Class” επιλεγμένο με το προκαθορισμένο όνομα κλάσης **helloworldapp.HelloWorldApp**
- Πατήστε “Finish”.



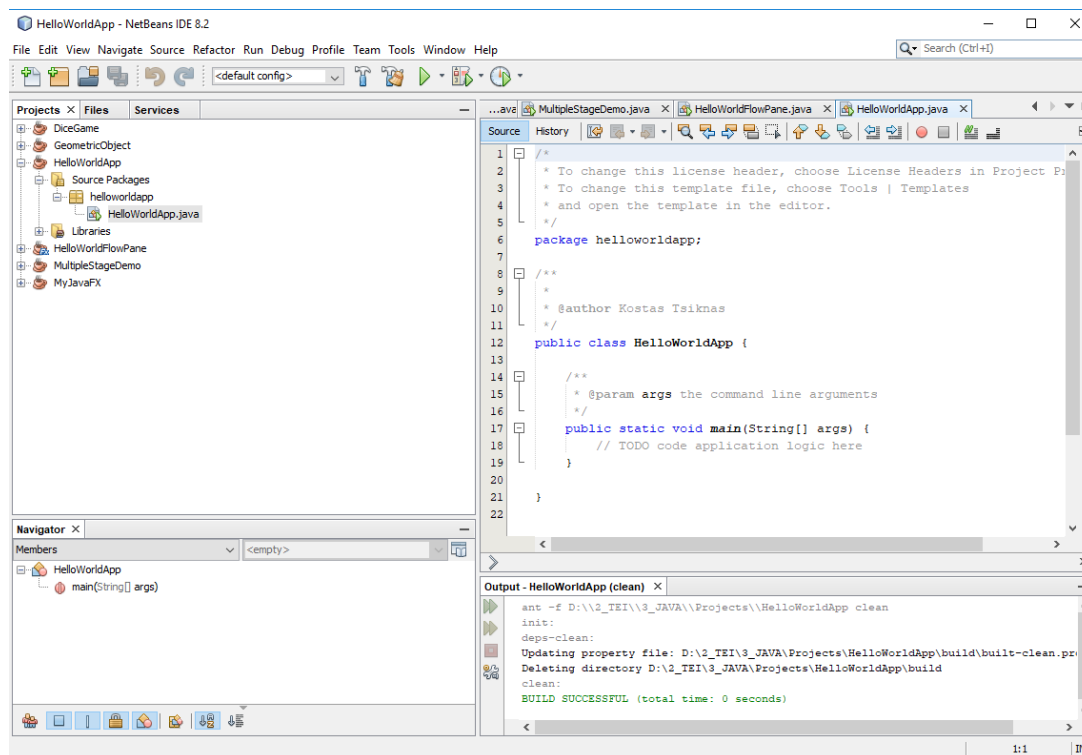
The screenshot shows the 'New Java Application' dialog box in NetBeans. The 'Steps' panel on the left indicates that step 2, 'Name and Location', is the current step. The 'Name and Location' section contains the following fields and options:

- Project Name:** HelloWorldApp
- Project Location:** D:\2_TEI\3_JAVA\Projects (with a 'Browse...' button)
- Project Folder:** D:\2_TEI\3_JAVA\Projects\HelloWorldApp
- ☐ Use Dedicated Folder for Storing Libraries (with a 'Libraries Folder:' field and a 'Browse...' button)
- ☒ Create Main Class helloworldapp.HelloWorldApp

At the bottom of the dialog, there are buttons for '< Back', 'Next >', 'Finish' (highlighted with a blue border), 'Cancel', and 'Help'.

Εισαγωγή στο Netbeans (3/6)

- Το project έχει δημιουργηθεί και έχει ανοιχθεί στο Netbeans IDE και πρέπει να βλέπετε τα ακόλουθα στοιχεία:
 - Το παράθυρο 'Projects', το οποίο περιέχει μια δενδροειδή δομή των στοιχείων του project, συμπεριλαμβανομένου των πηγαίων αρχείων, των βιβλιοθηκών που χρειάζεται ο κώδικας σας, κτλ.
 - Τον "Source Editor" με ένα το αρχείο HelloWorldApp ανοικτό.
 - Το παράθυρο 'Navigator', το οποίο μπορείτε να χρησιμοποιήσετε για να περιηγηθείτε γρήγορα μεταξύ των στοιχείων εντός του ανοιγμένου αρχείου.



Εισαγωγή στο Netbeans (4/6)

- Προσθέστε κώδικα στο project σας:

- Μέσα στην main αφαιρέστε την γραμμή:

`“// TODO code application logic here”`

```
public class HelloWorldApp {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        // TODO code application logic here  
    }  
}
```

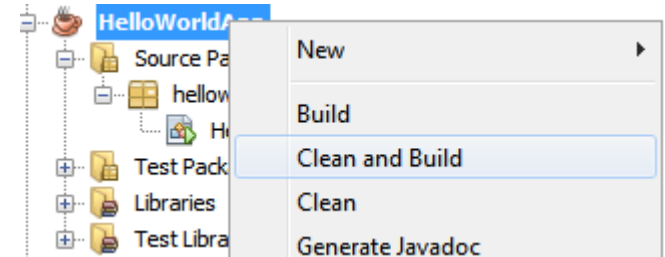
- Στην θέση της πληκτρολογήστε: `“ System.out.println(“Hello World”); ”`

```
public class HelloWorldApp {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        System.out.println("Hello World");  
    }  
}
```

Εισαγωγή στο Netbeans (5/6)

Κάντε compile το project:

- Στο παράθυρο “projects” στον τίτλο του project σας πατήστε δεξί click → Clean and Build

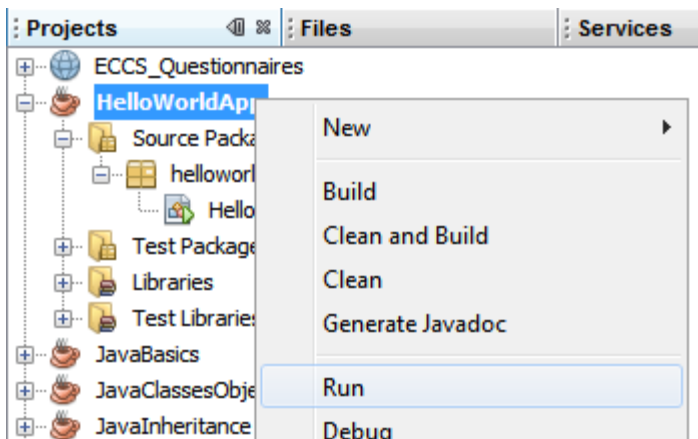


```
ant -f D:\2_TEI\3_JAVA\Projects\HelloWorldApp -Dnb.internal.action.name=rebuild clean jar
init:
deps-clean:
Created dir: D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build
Updating property file: D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build\build-clean.properties
Deleting directory D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build
clean:
init:
deps-jar:
Created dir: D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build
Updating property file: D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build\build-jar.properties
Created dir: D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build\classes
Created dir: D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build\empty
Created dir: D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build\generated-sources\ap-source-output
Compiling 1 source file to D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build\classes
compile:
Created dir: D:\2_TEI\3_JAVA\Projects\HelloWorldApp\dist
Copying 1 file to D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build
Nothing to copy.
Building jar: D:\2_TEI\3_JAVA\Projects\HelloWorldApp\dist\HelloWorldApp.jar
To run this application from the command line without Ant, try:
java -jar "D:\2_TEI\3_JAVA\Projects\HelloWorldApp\dist\HelloWorldApp.jar"
jar:
BUILD SUCCESSFUL (total time: 1 second)
```

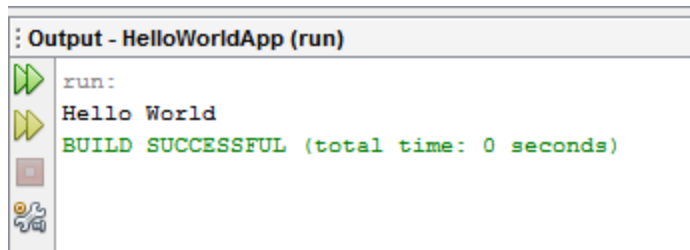
Αν το παράθυρο “Output” εμφανίσει **BUILD SUCCESSFUL** αυτό σημαίνει ότι το compile του project είναι επιτυχές και μπορεί να ακολουθήσει η εκτέλεση

Εισαγωγή στο Netbeans (6/6)

- Εκτελέστε το project σας:
 - Στο παράθυρο “projects” στον τίτλο του project σας πατήστε δεξί click
→ Run



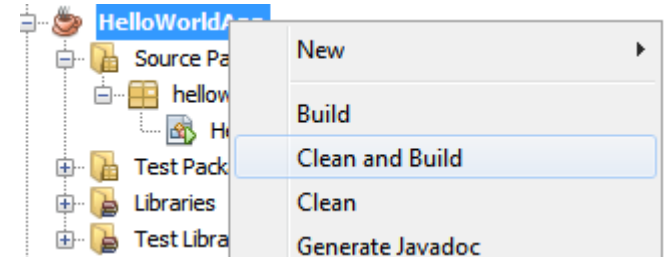
- Στο παράθυρο “Output” θα πρέπει να δείτε την πρόταση “Hello World!”



Εισαγωγή στο Netbeans (5/6)

Κάντε compile το project:

- Στο παράθυρο “projects” στον τίτλο του project σας πατήστε δεξί click → Clean and Build



```
ant -f D:\2_TEI\3_JAVA\Projects\HelloWorldApp -Dnb.internal.action.name=rebuild clean jar
init:
deps-clean:
Created dir: D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build
Updating property file: D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build\build-clean.properties
Deleting directory D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build
clean:
init:
deps-jar:
Created dir: D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build
Updating property file: D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build\build-jar.properties
Created dir: D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build\classes
Created dir: D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build\empty
Created dir: D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build\generated-sources\ap-source-output
Compiling 1 source file to D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build\classes
compile:
Created dir: D:\2_TEI\3_JAVA\Projects\HelloWorldApp\dist
Copying 1 file to D:\2_TEI\3_JAVA\Projects\HelloWorldApp\build
Nothing to copy.
Building jar: D:\2_TEI\3_JAVA\Projects\HelloWorldApp\dist\HelloWorldApp.jar
To run this application from the command line without Ant, try:
java -jar "D:\2_TEI\3_JAVA\Projects\HelloWorldApp\dist\HelloWorldApp.jar"
jar:
BUILD SUCCESSFUL (total time: 1 second)
```

Αν το παράθυρο “Output” εμφανίσει **BUILD SUCCESSFUL** αυτό σημαίνει ότι το compile του project είναι επιτυχές και μπορεί να ακολουθήσει η εκτέλεση

Περιεχόμενα ενότητας

- Εργαλεία ανάπτυξης εφαρμογών σε Java
 - Προαπαιτούμενα προγράμματα
 - Βασικά στοιχεία της JAVA
 - Δημιουργία πρώτου προγράμματος JAVA
 - Εισαγωγή στο Netbeans IDE
- Αντικειμενοστραφής προγραμματισμός σε Java
 - Η έννοια της κλάσης και του αντικειμένου
 - Δομητές (Constructors)
 - Υπερφόρτωση μεθόδων (overloading)
 - Κληρονομικότητα (Inheritance)
 - Πολυμορφισμός (polymorphism)
 - Αφηρημένες κλάσεις (abstract classes)

Η έννοια του αντικειμένου

- Στον πραγματικό κόσμο οτιδήποτε έχει μία υπόσταση
- Όλα έχουν καταστάσεις και συμπεριφορές



- π.χ «Το σκυλί» έχει καταστάσεις (όνομα, χρώμα, ράτσα) και συμπεριφορές (γαυγίζει, κουνάει την ουρά).
- Ο αντικειμενοστραφής προγραμματισμός αναφέρεται στον προγραμματισμό με χρήση αντικειμένων.
- Ένα αντικείμενο έχει μοναδική ταυτότητα, κατάσταση και συμπεριφορά

Ανάλυση αντικειμένου



Ποδήλατο

Καταστάσεις

(ιδιότητες ή χαρακτηριστικά αντικειμένου)

- Τρέχουσα Ταχύτητα (20 km/h)
- Τρέχουσα Σχέση ταχυτήτων (5^η)
- Τρέχων Ρυθμός πεταλιών (50 rpm)

Συμπεριφορές

(ενέργειες)

- Αλλαγή σχέσης ταχυτήτων
- Αλλαγή ρυθμού πεταλιών
- Πάτημα φρένου
- Επιτάγχνυση

Οι καταστάσεις ενός αντικειμένου μεταβάλλονται μέσω των συμπεριφορών του!

Ανάλυση αντικειμένου στον προγραμματισμό

- Κάθε αντικείμενο του πραγματικού κόσμου αναλύεται σε καταστάσεις και συμπεριφορές. Οι καταστάσεις συνήθως αλλάζουν μέσω των συμπεριφορών του αντικειμένου.
- Στον κόσμο του αντικειμενοστραφούς προγραμματισμού ένα αντικείμενο αποθηκεύει τις καταστάσεις του σε **πεδία δεδομένων (data fields)** και εκφράζει τις συμπεριφορές του μέσω των **μεθόδων (methods)**.
- Η αλλαγή τιμής μίας κατάστασης-μεταβλητής, μπορεί να γίνει κάνοντας χρήση μιας συμπεριφοράς-μεθόδου.

bicycle



data fields

- **int speed;**
- **int gear;**
- **int cadence;**

methods

- **setGear (int newValue)**
- **setCadence (int newValue)**
- **applyBreak (int decrement)**
- **speedUp (int increment)**

Η έννοια της κλάσης

- Κλάση είναι η δομή με την οποία περιγράφουμε αντικείμενα-οντότητες.
 - Με μία κλάση περιγράφουμε αντικείμενα-οντότητες τα οποία έχουν κοινά χαρακτηριστικά.
 - Παράδειγμα, αν θέλουμε να υλοποιήσουμε την κλάση Άνθρωπος περιγράφουμε τα κοινά χαρακτηριστικά που έχουν όλοι οι άνθρωποι μεταξύ τους.
- Σε μία κλάση δημιουργούμε όσα αντικείμενα επιθυμούμε.
- Είναι είναι μια γενική περιγραφή ενός αντικειμένου (ιδιοτήτων και ενεργειών του).
- Θα μπορούσαμε να παρομοιάσουμε την κλάση σαν:
 - α) *Συνταγή ενός γλυκού :*

Μπακλαβές - Περιεχόμενο

- 2 φλιτζάνια νερό
- 3 φλιτζάνια ζάχαρη
- λίγη κανέλα
- 1/2 φλιτζάνι βούτυρο
- 6 φλιτζάνια καρυδόψυχα
- 1 κουτί φύλλο
- φλούδα λεμονιού
- 1 ξυλάκι κανέλα

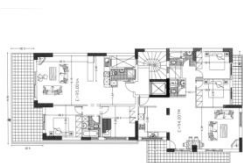
από την οποία
δημιουργούμε



το γλυκό



– β) Αρχιτεκτονικό σχέδιο:



από το οποίο
δημιουργούμε



το σπίτι



Δομή κλάσης

`class Bicycle {` → 1. Όνομα Κλάσης

```
    int cadence;  
    int speed;  
    int gear;
```

→ 2. Πεδία δεδομένων
(data fields)

```
Bicycle() {  
    cadence = 0;  
    speed = 0;  
    gear = 1;  
}
```

```
Bicycle(int startCadence, int startSpeed, int startGear) {  
    cadence = startCadence;  
    speed = startSpeed;  
    gear = startGear;  
}
```

→ 3. Constructors

```
int getCadence() {  
    return cadence;  
}
```

```
int getSpeed() {  
    return speed;  
}
```

```
int getGear() {  
    return gear;  
}
```

→ 4. Getters

```
void setCadence(int newValue) {  
    cadence = newValue;  
}
```

```
void setSpeed(int newValue) {  
    speed = newValue;  
}
```

```
void setGear(int newValue) {  
    gear = newValue;  
}
```

→ 5. Setters

```
void speedUp(int increment) {  
    speed = speed + increment;  
}
```

```
void applyBrakes(int decrement) {  
    speed = speed - decrement;  
}
```

→ 6. Μέθοδοι –Συμπεριφορές

Δομητές - Constructors

- Ο Constructor μοιάζει με μία μέθοδο, με την διαφορά ότι έχει ως όνομα το όνομα της κλάσης και δεν έχει τιμή επιστροφής.
- Ο Constructor είναι αυτός που καλείται όταν δημιουργούμε ένα αντικείμενο.
- Ο Constructor συνήθως αρχικοποιεί τα πεδία του αντικειμένου.
- Σε μία κλάση μπορούμε να έχουμε περισσότερους του ενός Constructors αρκεί να έχουν διαφορετική λίστα παραμέτρων (constructor overload).

```
Bicycle() {  
    cadence = 0;  
    speed = 0;  
    gear = 1;  
}
```

```
Bicycle(int startCadence, int startSpeed, int startGear) {  
    cadence = startCadence;  
    speed = startSpeed;  
    gear = startGear;  
}
```

Getters - Setters

- *Getters*

- Getter είναι μία μέθοδος με την οποία διαβάζουμε την τιμή ενός πεδίου μίας κλάσης

```
int getGear() {  
    return gear;  
}
```

- *Setters*

- Setter είναι μία μέθοδος με την οποία αλλάζουμε την τιμή ενός πεδίου μίας κλάσης

```
void setGear(int newValue) {  
    gear = newValue;  
}
```

Υπερφόρτωση μεθόδων - Method Overload

- Σε μία κλάση μπορούμε να έχουμε περισσότερες από μια μεθόδους με το ίδιο όνομα αρκεί να έχουν διαφορετική λίστα παραμέτρων

```
public class Bicycle {  
  
    int cadence;  
    int speed;  
    int gear;  
  
    void speedUp(int increment) {  
        speed = speed + increment;  
    }  
  
    void speedUp() {  
        speed += 10;  
    }  
  
    void speedUp(int increment, int maxSpeed) {  
        if ((speed + increment) < maxSpeed) {  
            speed += increment;  
        } else {  
            speed = maxSpeed;  
        }  
    }  
}
```

Κανόνες ορθής σύνταξης

- Όνομα κλάσης άρα και constructor ξεκινάει με κεφαλαίο.
- Όνομα πεδίου, μεταβλητής ή μεθόδου ξεκινάει με μικρό.
- Η πρώτη λέξη στο όνομα μεθόδου είναι ρήμα.
- Εάν ένα όνομα αποτελείται από περισσότερες από μία λέξεις, τότε κάθε επόμενη λέξη ξεκινά με κεφαλαίο.
- Στην σύνταξη κλάσεων ακολουθούμε την σειρά
 1. Πεδία
 2. Constructors
 3. Getters
 4. Setters
 5. Μέθοδοι

Δημιουργία αντικειμένων

- Τα αντικείμενα είναι στιγμιότυπα της κλάσης
- Η δημιουργία αντικειμένου έχει σαν αποτέλεσμα τη κλήση του constructor
- Η επιλογή του constructor καθορίζεται από το πλήθος των ορισμάτων (overload)

```
public static void main(String[] args) {
```

```
    Bicycle myBike = new Bicycle(20,15,5);
```

```
    Bicycle yourBike = new Bicycle();
```

```
    System.out.println(myBike.getGear());
```

```
    System.out.println(yourBike.getGear());
```

```
}
```

Με την εκτέλεση του πρώτου μισού της εντολής (Bicycle myBike) γίνεται δέσμευση χώρου μνήμης για μια μεταβλητή τύπου Bicycle.

Με την εκτέλεση του δεύτερου μισού της εντολής (myBike = new Bicycle(20,15,5);) δημιουργείται ένα στιγμιότυπο του Bicycle και εκχωρείται στην μεταβλητή myBike η οποία είναι τύπου Bicycle.

Με την εκτέλεση της εντολής myBike.getGear(); Καλούμε την μέθοδο getGear() της κλάσης Bicycle και τυπώνουμε την τιμή που επιστρέφει.

Access modifiers

- Καθορίζουν το επίπεδο προσβασιμότητας μιας μεταβλητής ή μιας μεθόδου μίας κλάσης από άλλες κλάσεις.
 - public → η μεταβλητή / μέθοδος είναι προσβάσιμη από όλες τις κλάσεις
 - private → η μεταβλητή / μέθοδος είναι προσβάσιμη μόνο μέσα στην ίδια την κλάση

```
public class Bicycle {
```

```
    private int gear;  
    public int speed;
```

```
    public Bicycle() {  
        gear = 1;  
        speed = 0;  
    }
```

```
    public int getGear() {  
        return gear;  
    }
```

```
    private int getSpeed() {  
        return speed;  
    }
```

```
public class Main {
```

```
    public static void main(String[] args) {  
        Bicycle myBike = new Bicycle();
```

```
        System.out.println(myBike.speed);  
        System.out.println(myBike.getSpeed() );
```

```
        System.out.println(myBike.gear);  
        System.out.println(myBike.getGear() );
```

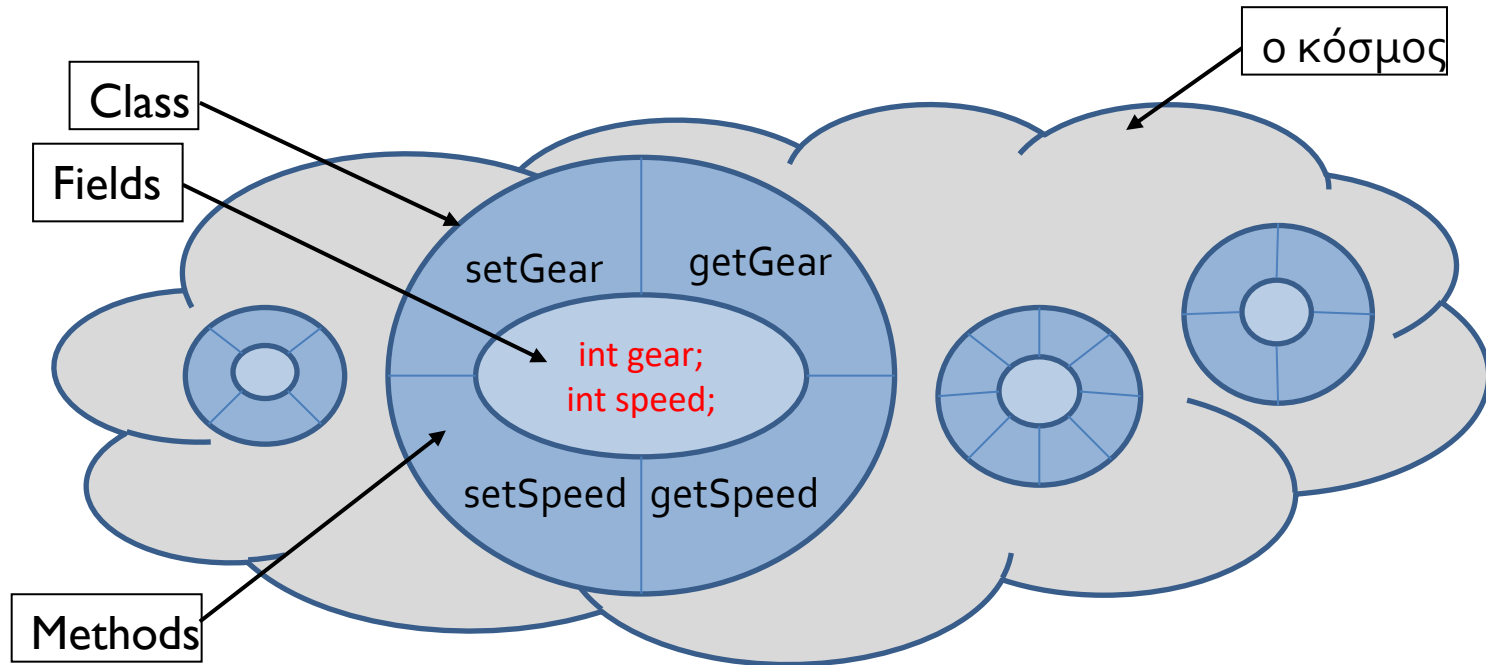
```
    }
```

```
}
```



Ενθυλάκωση (encapsulation)

- Η τεχνική κατά την οποία αποκρύπτεται η εσωτερική κατάσταση της κλάσης και η αλληλεπίδραση μαζί της γίνεται μέσω των μεθόδων.
- Δηλώνουμε τα **fields** ως **private** και κάνουμε προσπέλαση (ανάγνωση ή αλλαγή τιμής) στις μεταβλητές μιας κλάσης μέσω μεθόδων(π.χ. **setters**, **getters**).



Instance and Class Members (1/3)

- **Instance Variable/Method** → Κάθε αντικείμενο της κλάσης έχει το δικό του στιγμιότυπο (αντίγραφο) αυτής της μεταβλητής/μεθόδου.
- **Class Variable/Method** → Όλα τα αντικείμενα της κλάσης έχουν κοινή αυτήν την μεταβλητή/μέθοδο.
- **Για παράδειγμα:** Θέλουμε σε κάθε ποδήλατο να κρατάμε ένα μοναδικό σειριακό αριθμό. Στο πρώτο ποδήλατο αυτός θα είναι το 1 στο δεύτερο το 2 κ.ο.κ.
 - Το κάθε ποδήλατο λοιπόν έχει τον δικό του σειριακό αριθμό και γι' αυτό είναι instance variable

```
private int id ;
```

- Θα πρέπει όμως να κρατάμε κάπου τον αριθμό των ποδηλάτων που έχει δημιουργηθεί για να ξέρουμε ποιόν σειριακό αριθμό να δώσουμε στον επόμενο. Έτσι το κρατάμε σε μια class variable

```
private static int numberOfBicycles;
```


Instance and Class Members (2/3)

```
public class Bicycle {  
    //    instance variable  
    private int id;  
    //    class variable  
    private static int numberOfBicycles = 0 ;  
    //    class constructor  
    public Bicycle() {  
        id = ++numberOfBicycles;  
    }  
    //    instance method - getter  
    public int getId() {  
        return id;  
    }  
    //    class method - getter  
    public static int getNumberOfBicycles() {  
        return numberOfBicycles;  
    }  
    //    instance method - setter  
    public void setId(int id) {  
        this.id = id;  
    }  
    //    class method - setter  
    public static void setNumberOfBicycles(int numberOfBicycles) {  
        Bicycle.numberOfBicycles = numberOfBicycles;  
    }  
}
```

Η κλάση Bicycle
με instance και
class members

Instance and Class Members (3/3)

```
public class Main {  
  
    public static void main(String[] args) {  
        Bicycle myBike = new Bicycle();  
        Bicycle yourBike = new Bicycle();  
  
        System.out.println("myBike`s Id: " + myBike.getId());  
        System.out.println("yourBike`s Id: " + yourBike.getId());  
  
        System.out.println("Number of Bicycles: "+Bicycle.getNumberOfBicycles());  
    }  
}
```

Η main όπου
κάνουμε κλήσεις τόσο
των instance
methods όσο και των
class methods

```
Output - JavaInheritance (run)  
run:  
myBike`s Id: 1  
yourBike`s Id: 2  
Number of Bicycles: 2  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Το output αφού
τρέξουμε την
παραπάνω main

Κληρονομικότητα – Inheritance (1/3)

- Κάποιες κλάσεις αντικειμένων έχουν κοινά χαρακτηριστικά με άλλες κλάσεις
- **Κληρονομικότητα** είναι η δυνατότητα της κλάσης να κληρονομεί καταστάσεις και συμπεριφορές από μία άλλη
- Μια κλάση που κληρονομεί τα χαρακτηριστικά μια άλλης κλάσης (**υπερκλάσης**) ονομάζεται **υποκλάση** της κλάσης αυτής
- **Κάθε κλάση περιέχει όλα τα χαρακτηριστικά της υπερκλάσης της και κάποια επιπλέον**
- Κάθε κλάση μπορεί να έχει πολλές υποκλάσεις αλλά μόνο μια υπερκλάση
- Έτσι δημιουργείται η ιεραρχία της Java στην κορυφή της οποίας βρίσκεται η κλάση **Object**
- **Η δήλωση ότι μια κλάση κληρονομεί από κάποια άλλη γίνεται με τον τελεστή **extends****

Κληρονομικότητα – Inheritance (2/3)

Bicycle



Υπερκλάση των
κλάσεων Mountain
Bike, Road Bike
και Tandem Bike

Mountain Bike



Υποκλάση της
κλάσης Bicycle

Road Bike



Υποκλάση της
κλάσης Bicycle

Tandem Bike



Υποκλάση της
κλάσης Bicycle

Κληρονομικότητα – Inheritance (3/3)

Bicycle

```
public class Bicycle {  
    private int gear;  
    private int speed;  
  
    public Bicycle() {  
        gear = 1;  
        speed = 0;  
    }  
  
    public int getGear() {  
        return gear;  
    }  
  
    public int getSpeed() {  
        return speed;  
    }  
}
```

extends

Mountain Bike

```
public class MountainBike extends Bicycle{  
    private int seatHeight;  
    ...  
}
```

Road Bike

```
public class RoadBike extends Bicycle{  
    private int tireWidth;  
    ...  
}
```

Tandem Bike

```
public class TandemBike extends Bicycle{  
    private int numberOfSeats;  
    ...  
}
```

Οι τελεστές super / this

- **super** → Χρησιμοποιείται μέσα σε μία κλάση και μας δίνει πρόσβαση στα πεδία, τους κατασκευαστές ή τις μεθόδους της υπερκλάσης
- **this** → Χρησιμοποιείται μέσα σε μία κλάση και μας δίνει πρόσβαση στα πεδία, τους κατασκευαστές ή τις μεθόδους της ίδιας της κλάσης

```
public class MountainBike extends Bicycle{
```

```
    private int seatHeight;
```

```
    MountainBike() {
```

```
        super();
```

```
        seatHeight = 1;
```

```
    }
```

```
    public int getSeatHeight() {
```

```
        return seatHeight;
```

```
    }
```

```
    public void setSeatHeight(int seatHeight) {
```

```
        this.seatHeight = seatHeight; This: Κλήση δεδομένων της κλάσης
```

```
    }
```

```
    @Override
```

```
    public void printDescription() {
```

```
        super.printDescription();
```

```
        System.out.println("Seat Height: " + this.getSeatHeight());
```

```
    }
```

*super: Κλήση του constructor της υπερκλάσης
(η χρήση του επιτρέπεται μόνο εντός του constructor)*

*Πολυμορφισμός (Method overload):
Υπερφόρτωση constructor*

super: Κλήση μεθόδου της υπερκλάσης

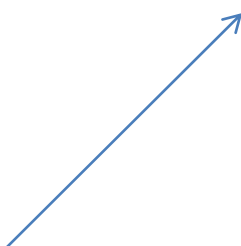
Πολυμορφισμός - Μέθοδοι υλοποίησης

- **Ορισμός:** Η ιδιότητα των υποκλάσεων τόσο να ορίζουν δικές τους συμπεριφορές όσο και ταυτόχρονα να διατηρούν κάποιες άλλες όπως έχουν οριστεί στις υπερκλάσεις τους.
- **Υπερφόρτωση μεθόδου (method overloading):** σε μια κλάση ορίζονται 2 ή περισσότερες μέθοδοι με το ίδιο όνομα και διαφορετικές παραμέτρους – διαφορετικό πρωτότυπο (προηγούμενο παράδειγμα)
- **Επανακαθορισμός μεθόδου (method overriding):** Η μέθοδος μιας βασική κλάσης ορίζεται ξανά στην παραγόμενη κλάση με νέα υλοποίηση και το ίδιο πρωτότυπο
- **Αφηρημένη κλάση (abstract class):** Μια κλάση περιέχει τουλάχιστον μία μέθοδο που ορίζεται, αλλά δεν υλοποιείται. Η υλοποίηση της κλάσης γίνεται στις παραγόμενες μεθόδους.

Παράδειγμα υλοποίησης πολυμορφισμού με επανακαθορισμό μεθόδων (method override)

- Ο τελεστής `@Override` χρησιμοποιείται για τη δήλωση της μεθόδου στην παραγόμενη κλάση και την υλοποίησή της με διαφορετικό τρόπο.

```
public class Bicycle {  
    private int gear;  
    private int speed;  
  
    public Bicycle() {  
        gear = 1;  
        speed = 0;  
    }  
  
    public int getGear() {  
        return gear;  
    }  
  
    public int getSpeed() {  
        return speed;  
    }  
  
    public void printDescription() {  
        System.out.println("Gear: " + gear);  
        System.out.println("Speed: " + speed);  
    }  
}  
  
public class MountainBike extends Bicycle{  
    private int seatHeight;  
  
    public int getSeatHeight() {  
        return seatHeight;  
    }  
  
    public void setSeatHeight(int seatHeight) {  
        this.seatHeight = seatHeight;  
    }  
  
    @Override  
    public void printDescription() {  
        System.out.println("Gear: " + this.getGear());  
        System.out.println("Speed: " + this.getSpeed());  
        System.out.println("Seat Height: " + this.getSeatHeight());  
    }  
}
```



- ❖ Η διαδικασία αυτή λέγεται “**method overriding**” και δίνει την δυνατότητα σε μία κλάση να διαφοροποιεί μία μέθοδο που κληρονομεί από την υπερκλάση της, διατηρώντας ίδια την υπογραφή της (όνομα μεθόδου, αριθμός και τύπος παραμέτρων)

Αφηρημένη κλάση (abstract class)

- Μια κλάση ονομάζεται αφηρημένη όταν περιέχει τουλάχιστον μια αφηρημένη μέθοδο που δεν περιλαμβάνει υλοποίηση
 - Η κλάση δηλώνεται **public abstract class MyAbstractClass**
 - με τη μέθοδο **public abstract returnType methodName();**
- Η αφηρημένη κλάση λειτουργεί σαν καλούπι για την κατασκευή παραγόμενων που προσφέρουν εναλλακτικές υλοποιήσεις στις αφηρημένες μεθόδους....
- Η δημιουργία αντικειμένων αφηρημένης κλάσης δεν επιτρέπεται από τον compiler
- Η μη υλοποίηση αφηρημένων μεθόδων δεν επιτρέπεται από τον compiler

Παράδειγμα Αφηρημένης κλάσης

```
package bicycle;
```

```
public abstract class Bicycle {
```

```
    private int gear;
```

```
    public int speed;
```

```
    private static int numOfBikes = 0;
```

```
    public Bicycle() {
```

```
        gear = 1;
```

```
        speed = 20;
```

```
        numOfBikes++;
```

```
    }
```

```
    public int getGear() {
```

```
        return gear;
```

```
    }
```

```
    public int getSpeed() {
```

```
        return speed;
```

```
    }
```

```
    public static int getNumOfBikes() {
```

```
        return numOfBikes;
```

```
    }
```

```
    public void printDescription() {
```

```
        System.out.println("Gear: " + gear);
```

```
        System.out.println("Speed: " + speed);
```

```
    }
```

```
    public abstract String getSuspensionType();
```

```
    public abstract void setSuspensionType(String suspType
```

```
}
```

Παραγόμενη κλάση Circle

```
public class MountainBike extends Bicycle{
```

```
    private int seatHeight;
```

```
    private String suspensionType;
```

```
    MountainBike() {
```

```
        super();
```

```
        seatHeight = 15;
```

```
    }
```

```
    public int getSeatHeight() {
```

```
        return seatHeight;
```

```
    }
```

```
    public void setSeatHeight(int seatHeight) {
```

```
        this.seatHeight = seatHeight;
```

```
    }
```

```
    @Override
```

```
    public void printDescription() {
```

```
        super.printDescription();
```

```
        System.out.println("Seat Height: " + seatHeight);
```

```
    }
```

```
    public String getSuspensionType() {
```

```
        return suspensionType;
```

```
    }
```

```
    public void setSuspensionType(String suspensionType) {
```

```
        this.suspensionType = suspensionType;
```

```
    }
```

```
}
```

Κλήση παραγόμενης κλάσης που κάνει χρήση αφηρημένης κλάσης

```
import bicycle.Bicycle;

/**
 *
 * @author Kostas Tsiknas
 */
public class TestBicycle {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        // Bicycle bike1 = new Bicycle();
        // Bicycle bike2 = new Bicycle();
        Bicycle bike3 = new MountainBike();

        //System.out.println(bike1.getSpeed());
        //System.out.println(Bicycle.getNumOfBikes());
        // bike2.printDescription();
        bike3.printDescription();
        bike3.setSuspensionType("hard");
        System.out.println("Suspension type: " + bike3.getSuspensionType());
    }
}
```

ΔΕΝ ΕΠΙΤΡΕΠΕΤΑΙ Η ΔΗΜΙΟΥΡΓΙΑ
ΣΤΙΓΜΙΟΤΙΠΟΥ ΑΦΗΡΗΜΕΝΗΣ ΚΛΑΣΗΣ

ΟΙ ΜΕΘΟΔΟΙ ΤΗΣ
ΠΑΡΑΓΟΜΕΝΗΣ
ΚΛΑΣΗΣ
ΚΑΛΟΥΝΤΑΙ
ΚΑΝΟΝΙΚΑ

Χρήσιμα Links (1/3)

- **The Java Tutorials - Trail: Getting Started**

- <http://download.oracle.com/javase/tutorial/getStarted/index.html>
- *«This trail provides everything you'll need to know about getting started with the Java programming language.»*

- **The Java Tutorials - Trail: Learning the Java Language**

- **Lesson: Language Basics**

- <http://download.oracle.com/javase/tutorial/java/nutsandbolts/index.html>
- *«Language Basics describes the traditional features of the language, including variables, arrays, data types, operators, and control flow.»*

Χρήσιμα Links (2/3)

- **The Java Tutorials - Trail: Learning the Java Language**
 - **Lesson: Object-Oriented Programming Concepts**
 - What Is an Object?
 - What Is a Class?
 - <http://download.oracle.com/javase/tutorial/java/concepts/index.html>
 - *“Object-Oriented Programming Concepts teaches you the core concepts behind object-oriented programming: objects, messages, classes, and inheritance. This lesson ends by showing you how these concepts translate into code. Feel free to skip this lesson if you are already familiar with object-oriented programming.”*
- **The Java Tutorials - Trail: Learning the Java Language**
 - **Lesson: Classes and Objects**
 - <http://download.oracle.com/javase/tutorial/java/javaOO/index.html>
 - *“Classes and Objects describes how to write the classes from which objects are created, and how to create and use the objects.”*

Χρήσιμα Links (3/3)

- **The Java Tutorials - Trail: Learning the Java Language**
 - **Lesson: Object-Oriented Programming Concepts**
 - What Is Inheritance?
 - <http://download.oracle.com/javase/tutorial/java/concepts/index.html>
 - *“Object-Oriented Programming Concepts teaches you the core concepts behind object-oriented programming: objects, messages, classes, and inheritance. This lesson ends by showing you how these concepts translate into code. Feel free to skip this lesson if you are already familiar with object-oriented programming.”*
- **The Java Tutorials - Trail: Learning the Java Language**
 - **Lesson: Interface and Inheritance**
 - Inheritance
 - <http://download.oracle.com/javase/tutorial/java/landl/subclasses.html>
“This section describes the way in which you can derive one class from another. That is, how a subclass can inherit fields and methods from a superclass. You will learn that all classes are derived from the Object class, and how to modify the methods that a subclass inherits from superclasses.”