UoM-Applied-Informatics/s3/object_oriented_programming/lectures at main · iosifidis/UoM-Applied-Informatics

github.com/iosifidis/UoM-Applied-Informatics/tree/main/s3/object_oriented_programming/lectures

iosifidis

README.md

Προγράμματα διαλέξεων

Σε κάθε φάκελο, μπορείτε να δείτε τα αρχεία με τον κώδικα αλλά και ένα αρχείο zip για να το εισάγετε ως Project στο eclipse.

- Products_(notOO): Πρόγραμμα που υπολογίζει ποιο είναι το καλύτερο προϊόν. Παράδειγμα ως διαδικαστικός προγραμματισμός.
- ProductsWithClasses: Το ίδιο πρόγραμμα με το 1. Εδώ δημιουργείται μια κλάση Product και μεταφέρονται-μετατρέπονται σε μεθόδους, όλες οι λειτουργίες που βρισκόταν μέσα στην main.
- University_2_11: Πρόγραμμα φοιτητών. Χρήση πολλών constructors και χρήση μεθόδων set (θέτω τιμή) και get (κάνω ανάγνωση τιμής).
- University_9_11: Πρόγραμμα με φοιτητές. Καταχώρηση σε πίνακα με μαθήματα και εκτύπωση πίνακα με φοιτητές που έχουν εγγραφεί στο μάθημα.
- University_9_11_v2: Εργασία με ArrayList με φοιτητές, μαθήματα, αίθουσες και γραμματεία. Δημιουργία φοιτητών, μαθημάτων και αιθουσών. Ορισμός αιθουσών για κάθε μάθημα. Εγγραφή φοιτητών σε μαθήματα και στην συνέχεια εκτύπωση λεπτομερειών των μαθημάτων. Δημιουργία αντικειμένου γραμματείας και καταχώρηση μαθημάτων. Εκτύπωση πληροφοριών γραμματείας.
- University_16_11: Αρχή της υποκατάστασης και πολυμπορφισμός.
- Timestamp: Τήρηση αναλοίωτων (πχ συνθήκη ώρας 0<= hour < 24).
- Game: Ορισμός στατικής ιδιότητας μέσα στην κλάση ώστε να είναι "ορατή" σε όλα τα αντικείμενα (πχ μετρητής σε στρατιωτάκια). Επίσης στατική μέθοδος για να την καλώ, χωρίς να έχω φτιάξει αντικείμενο στην main. Ορισμός final σε μια ιδιότητα, δεν επιτρέπει σε κανέναν να αλλάξει την τιμή της.
- Bank: Κληρονομικότητα. Επαναορισμός ή επικάλυψη μεθόδου και protected ιδιότητες μιας υπερκλάσης για πρόσβαση από άλλες υποκλάσεις.
- University_23_11: Αφηρημένη (abstract) κλάση και αρχή της υποκατάστασης με τύπους φοιτητών. Επικάλυψη μεθόδου.
- Dataset: Χρήση των αφαιρέσεων (Measurable). Παράδειγμα με την έννοια της διασύνδεσης (Interface)
- GUI: Εισαγωγή στην γραφική διασύνδεση χρήστη
- University_7_12: Παράδειγμα υλοποίησης της toString και παράδειγμα σύγκρισης 2 αντικειμένων (με την equals)

- Students_7_12: Ολοκληρωμένο παράδειγμα με γραφική διεπαφή. Εισαγωγή φοιτητή με όνομα, ταυτότητα και μάθημα και εισαγωγή του σε λίστα φοιτητών (μέσα στην κλάση μάθημα) που πήραν το μάθημα. Στην συνέχεια εμφάνιση του μαθήματος και φοιτητών που πήραν το μάθημα. Επικοινωνία Γραφικής Διασύνδεσης με Κλάσεις Πεδίου προβλήματος.
- ChessBoard: Παράδειγμα ζωγραφικής ενός παραθύρου σκακιέρας.
- Containers_14_12: Παράδειγμα Containers and Ships. Ήταν θέμα εξετάσεων όπου προστέθηκε η γραφική διεπαφή. Δυο τύποι από containers που φορτώνονται σε πλοία. Το παράθυρο αποτελείται από 2 τύπους containers από 2 πλήκτρα (ανάλογα με το container) και επιλογή πλοίου. Στην εκτύπωση εμφανίζει και το σύνολο του κόστους. Η λίστα με τα πλοία περνάει ως παράμετρος από την Main στον κατασκευαστή του παραθύρου.
- DataStructures1_LinkedList: Εργασία με LinkedLists. Γνωριμία με το Iterator για να διατρέχω όλες τις δομές δεδομένων.
- DataStructures2_HashSet: Εργασία με σύνολα. Τα αποτελέσματα ΔΕΝ έχουν διάταξη και ΔΕΝ έχουν τα διπλότυπα.
- DataStructures3_Conversion: Πως προγραμματιστικά φτιάχνω μια ArrayList (περιέχει διλότυπα) και φτιάχνω ένα Collection με μια HashSet και εισάγω μέσα την ArrayList για να εξάγω τα διπλότυπα.
- DataStructures4_SetOperations: Πράξεις συνόλων.
- DataStructures5_TreeSet: Δημιουργία ενός TreeSet (δέντρου) και στην εκτύπωση παρατηρούμε ότι τα αποτελέσματα είναι ταξινομημένα.
- DataStructures6_TreeSetComparable: Καταχωρώ σε ένα TreeSet αντικείμενα. Όμως βγάζει σφάλμα διότι δεν ξέρει πως να τα ταξινομήσει. Οπότε φτιάχνω μια διασύνδεση Comparable και υλοποιώ την compareTo για να γίνει η ταξινόμηση των αντικειμένων. Υλοποίηση εσωτερικών συγκριτών.
- DataStructures7_TreeSetComparator: Καταχώρηση σε ένα TreeSet αντικειμένων. Υλοποίηση 2 εξωτερικών συγκριτών.
- DataStructures8_HashSet: Καταχώρηση σε ένα HashSet 2 ίδιων αντικειμένων. Πρέπει να επικαλύψω την hashcode και equals για να μην καταχωρεί τα 2 ίδια αντικείμενα (πρέπει να ορίσω πχ να καταχωρούνται βάση του id).
- DataStructures9_Algorithms: Δημιουργία ενός Collection ArrayList. Εύκολος τρόπος για ταξινόμηση, αντίστροφη ταξινόμηση, τυχαίο ανακάτεμα, εναλλαγή σειράς εύρεση συχνότητας ευμφάνισης και ελάχιστου-μέγιστου επί της ArrayList.
- DataStructures10_GUI: Γραφική διεπαφή με μοντέλο λίστας στην οποία προτίθεται ένα γραφικό συστατικό μπάρα κύλισης. Ο χειρισμός συμβάντων των κουμπιών υλοποιειται μέσα στην main, κάνοντας σύμπτυξη των 4 βημάτων.
- DataStructures12_HashMap: Εισαγωγή στα hashmaps.
- LibraryMapsQ: Απλή υλοποίηση προγράμματος βιβλιοθήκης με την χρήση των hashmaps.
- Charts: Εισαγωγή εξωτερικών βιβλιοθηκών (Properties>Java build path>Libraries>Add external jars). Κατασκευή παραθύρου με διαγράμματα.
- Chess: Συμπλήρωση της σκακιέρας με ένα πιόνι που μετακινείται με κλικ του ποντικιού.
- Files_v1: Τρόπος εγγραφής σε αρχείο κειμένου (ascii).

- Files_v2: Επέκταση του Files_v1 για ανάγνωση. Άνοιγμα μιας γραφικης διεπαφής με πλήκτρο open file. Ανοίγει επιλογέας αρχείου και στη συνέχεια διαβάζεται το αρχείο γραμμή-γραμμή. Το αποτέλεσμα, εκτυπώνεται στο τερματικό.
- BinaryFiles_v1: Τρόπος εγγραφής αντικειμένου σε δυαδικό αρχείο.
- BinaryFiles_v2: Τρόπος ανάγνωσης αντικειμένου από δυαδικό αρχείο.
- BinaryFiles_v3: Τρόπος εγγραφής μιας ArrayList με αντικείμενα σε ένα δυαδικό αρχείο.
- BinaryFiles_v4: Τρόπος ανάγνωσης μιας ArrayList με αντικείμενα από ένα δυαδικό αρχείο.
- BinaryFiles_v5: Τρόπος εγγραφής μιας ArrayList που έχουν αποθηκευτεί με πληροφορίες αντικειμένων που είναι συνδεδεμένα με αντικείμενα (υπάλληλοι με αυτοκίνητα), σε ένα δυαδικό αρχείο.
- BinaryFiles_v6: Τρόπος ανάγνωσης μιας ArrayList που έχει αποθηκευτεί με πληροφορίες αντικειμένων που είναι συνδεδεμένα με αντικείμενα (υπάλληλοι με αυτοκίνητα), σε ένα δυαδικό αρχείο.

```java
import java.util.Scanner;


public class Main {

 /**
  * @param args
  */
 public static void main(String[] args) {

  Scanner in = new Scanner(System.in);

  //κατασκευή αντικειμένου της κλάσης Product
  Product best = new Product();

  boolean more = true;
  while(more) {

   Product current = new Product();

   //κλήση λειτουργίας ή μεθόδου readData επί του αντικειμένου current
   //Αποστολή του μηνύματος readData στο αντικείμενο current
   current.readData();

   //Ελεγχος αν το τρέχον είναι καλύτερο από το τρέχον καλύτερο, τότε άλλαξέ το
   if(current.isBetterThan(best))
    best=current;

   System.out.println("More data ? (1=YES, 2=NO)");
   int answer = in.nextInt();
   if(answer != 1)
    more = false;
   in.nextLine();

  }


  //Εκτύπωση καλύτερου προϊόντος
  System.out.println("BEST Product: ");
  best.printData();


 }

}
```

```java
import java.util.Scanner;

public class Product {

 // ιδιότητες κλάσης - attributes

 private String name;
 private int score;
 private double price;

 //κατασκευαστής - constructor
 public Product(){
  name = "";
  score = 0;
  price = 1;
 }

 // Λειτουργίες κλάσης -> μέθοδοι (methods)
 // Μέθοδος ανάγνωσης δεδομένων
 public void readData(){
  Scanner in = new Scanner(System.in);

  System.out.println("Please enter the product name: ");
  name = in.nextLine();
  System.out.println("Please enter the product price: ");
  price = in.nextDouble();
  System.out.println("Please enter the product score: ");
  score = in.nextInt();

 }

 // Μέθοδος εκτύπωσης δεδομένων
 public void printData(){

  System.out.println("The product is: " + name);
  System.out.println("The price is: " + price);
  System.out.println("The score is: " + score);
 }

 // Μέθοδος σύγκρισης δυο προϊόντων (2 αντικειμένων)
 public boolean isBetterThan(Product other){

  if(this.score/this.price > other.score/other.price)
   return true;
  return false;
 }


}
```

```java
// Φοιτητής: Ευάγγελος Ιωσηφίδης
// AM: iis21027

public class Main {

 public static void main(String[] args){

  // Μπορώ να πάρω τις τιμές από το πληκτρολόγιο
  // Scanner in = new Scanner(System.in);
  // System.out.println("Enter a Name: ");
  // String name = in.nextLine();
  // System.out.println("Enter some id: ");
  // String id = in.nextLine();
  // Student s1 = new Student(name,id);

  // χρήση διαφορετικών κατασκευαστών
  Student s1 = new Student("Babis", "ics20155");
  Student s2 = new Student("Maria");
  Student s3 = new Student();

  // εκτύπωση αντικειμένων
  s1.printInfo();
  s2.printInfo();
  s3.printInfo();

  // Θέτω το όνομα σε ένα αντικείμενο Student
  s3.setName("Takis");
  s3.printInfo();

  // Εκτυπώνω ΜΟΝΟ το όνομα του φοιτητή s1 λαμβάνοντάς το με την μέθοδο get
  System.out.println(s1.getName());

 }

}
```

```
public class Student{

 private String name;
 private String id;

 //Κατασκευαστής που δέχεται παραμέτρους όνομα και id
 public Student(String aName, String someId){
  this.name = aName;
  this.id = someId;

 }

 //Κατασκευαστής που δέχετε παράμετρο μόνο το όνομα
 public Student(String aName){
  name = aName;
  id = "not defined yet";
 }

 // κατασκευαστής χωρίς παραμέτρους
 public Student(){
  name = "not defined yet";
  id = "not defined yet";
 }

 // μέθοδος εκτύπωση πληροφοριών
 public void printInfo(){

  System.out.println("Name: " + name);
  System.out.println("ID: " + id);
 }

 // μέθοδος set θέτω τιμή στις ιδιότητες μέσω παραμέτρου
 public void setName(String aName){

  name = aName;
 }

 // μέθοδος get λαμβάνω μια τιμή από την ιδιότητα
 public String getName(){
  return name;

 }

}
```

```
public class Main {

 public static void main(String[] args){

  Student s1 = new Student("John", "ics20133");
  Student s2 = new Student("Mary", "iis19047");
  Student s3 = new Student("Babis", "iis20100");

  //s1.printInfo();  προσοχή, αν κληθεί εδώ η printInfo
  //s2.printInfo();  θα προκληθεί Null Pointer Exception

  Course c1 = new Course("Java" , 5);
  Course c2 = new Course("Machine Learning" , 5);


//  // Θέτω την τιμή του αντικειμένου μαθήματος c1 στην αναφορά s1
//  s1.setCourse(c1);
//
//  c1.printInfo();

   // Για να μην χρησιμοποιήσω αυτό, θα πάω στην Student και θα τροποποιήσω την setCourse
//  // Θέτω την τιμή του φοιτητή s1 στην αναφορά c1
//  c1.enrollStudent(s1);

//  s2.setCourse(c2); //σύνδεση Student S1->Course C1
//
//  s1.printInfo();
//  // s2.printInfo();


  // Γράφω 3 φοιτητές σε ένα μάθημα
  c1.enrollStudent(s1);
  c1.enrollStudent(s2);
  c1.enrollStudent(s3);

  // Εκτυπώνω όλους τους φοιτητές
  c1.printInfo();


 }
}
```

```
public class Student {

 private String name;
 private String id;
 // Ιδιότητα Course
 // Αναφορά προς το μάθημα
 private Course myCourse;

 // Μέθοδος set για ορισμό μαθήματος και εγγραφή φοιτητή στο μάθημα που παρακολουθεί
 public void setCourse(Course aCourse){

  myCourse = aCourse;
  // Μέθοδος που καλείται για να εγγράψει τον φοιτητή στο μάθημα
  aCourse.enrollStudent(this);
 }

 public Student(String aName, String someId){
  name = aName;
  id = someId;
 }


 public String getName() {
  return name;
 }


 public void setName(String name) {
  this.name = name;
 }


 public String getId() {
  return id;
 }


 public void setId(String id) {
  this.id = id;
 }

 public void printInfo(){

  System.out.println("Name: " + name);
  System.out.println("Id: " + id);
  System.out.println("---------------------");
  System.out.println("Is enrolled in ");
  if(myCourse!= null){
   System.out.println("Course: " + myCourse.getName());
     System.out.println("ECTS: " + myCourse.getECTS());
  }
  else
   System.out.println("No course selected");
  }

}
```

```
public class Course{

 private String name;
 private int ects;

// // Αναφορά που δείχνει σε έναν φοιτητή
// private Student enrolledStudent;

 // Για πολλούς φοιτητές που παρακολουθούν, πρέπει να φτιάξω έναν πίνακα
 private Student[] enrolledStudents = new Student[100];

 private int numberOfStudents;

 public Course(String aName, int aNumber){
  name = aName;
  ects = aNumber;
  // Η αρχικοποίηση είτε εδώ είτε στην δήλωσή του
  numberOfStudents=0;
 }

 // Μέθοδος set
 public void enrollStudent(Student s){
//  // Για έναν φοιτητή
//  enrolledStudent = s;

  // Για πολλούς φοιτητές

  enrolledStudents[numberOfStudents] = s;
  numberOfStudents++;

 }

 public void printInfo(){
  System.out.println("Course Name: " + name);
  System.out.println("ECTS: " + ects);
  System.out.println("has the following students");
  // Διατρέχω και εκτυπώνω τον πίνακα με τους φοιτητές
  for(int i=0; i<numberOfStudents; i++){
   System.out.println("student name: " + enrolledStudents[i].getName());

  }

//  // Εκτυπώνω το όνομα του φοιτητή
//  System.out.println("student name: " + enrolledStudent.getName());

 }

 public String getName(){
  return name;
 }

 public int getECTS(){
  return ects;
 }



}
```

```
public class Main {

 public static void main(String[] args) {

  Student s1 = new Student("Babis","ics20121");
  Student s2 = new Student("Maria", "iis19038");
  Student s3 = new Student("Sakis", "iis19039");

  Course c1 = new Course("Java", 5);
  Course c2 = new Course("Cryptography", 5);

  ClassRoom r1 = new ClassRoom("Amf12", 2);
  ClassRoom r2 = new ClassRoom("Erg234", 3);

  // Ορισμός αίθουσας στο μάθημα
  c1.setClassroom(r1);
  c2.setClassroom(r2);

  // Εγγραφή 3 φοιτητών στα ανάλογα μαθήματα
  c1.enrollStudent(s1);
  c1.enrollStudent(s2);
  c1.enrollStudent(s3);

  // Εκτύπωση των λεπτομερειών του μαθήματος
  c1.printCourseDetails();
  //c2.printCourseDetails();

  // Δημιουργία αντικειμένου γραμματείας
  Registry reg1 = new Registry("Applied Informatics");
  reg1.addCourse(c1); // Προσθήκη μαθήματος στην γραμματεία
  reg1.addCourse(c2); // Προσθήκη μαθήματος στην γραμματεία

  // Εκτύπωση δεδομένων της γραμματείας
  reg1.printDepartmentData();

 }

}
```

```
public class Student {

 private String name;
 private String id;

 // Κατασκευαστής
 public Student(String name, String id) {
  this.name = name;
  this.id = id;
 }

 // Μέθοδος εκτύπωσης
 public void printInfo(){

  System.out.println("Name: " + name);
  System.out.println("Id: " + id);
  System.out.println("--------------------");
 }

}
```

```java
import java.util.ArrayList;

public class Course {

 private String name;
 private int ects;
 // Φτιάχνω μια ArrayList για να καταχωρώ φοιτητές που έχουν εγγραφεί στο μάθημα
 // Βάζω <> και μέσα τον τύπο των αντικειμένων που θα αποθηκεύω στην ArrayList
 private ArrayList<Student> students = new ArrayList<>();
 // Ιδιότητα ClassRoom για σύνδεση μαθήματος με τάξη
 private ClassRoom myClassroom;

 // Κατασκευαστής
 public Course(String name, int ects) {
  this.name = name;
  this.ects = ects;
 }

 // Μέθοδος εγγραφής στο μάθημα
 public void enrollStudent(Student aStudent){

  // Κάνω έλεγχο αν το μέγεθος της λίστα των μαθητών είναι μικρότερο από το μέγεθος που χωράει η αίθουσα
  if(students.size() < myClassroom.getCapacity()){
   students.add(aStudent); // Πρόσθεσε τον φοιτητή
   System.out.println("Student successfully enrolled"); // Εκτύπωσε μήνυμα επιτυχίας
  }
  else
   System.out.println("Sorry the course is full"); // Εκτύπωσε μήνυμα αποτυχίας
 }

 // Μέθοδος εκτύπωσης λεπτομερειών μαθήματος
 public void printCourseDetails(){
  System.out.println("Course name: " + name);
  System.out.println("ECTS: " + ects);
  // Εκτύπωση αίθουσας
  System.out.println("Classroom: " + myClassroom.getLocation());

  for(int i=0 ; i<students.size(); i++){
   // Αν στο ArrayList δεν είχα εισάγει το <>, τότε πρέπει να κάνω ρητή μετατροπή σε Student
   // Student s = (Student) students.get(i);

   Student s = students.get(i);
   s.printInfo();

   // Σε μια σειρά θα έβγαινε students.get(i).printInfo();
  }

 }

 public void setClassroom(ClassRoom aClassroom) {

  myClassroom = aClassroom;
 }

}
```

```
public class ClassRoom {

 private String location;
 private int capacity;

 // Κατασκευαστής
 public ClassRoom(String location, int capacity) {
  this.location = location;
  this.capacity = capacity;
 }

 public int getCapacity() {
  return capacity;
 }

 public String getLocation() {
  return location;
 }


}
```

```
import java.util.ArrayList;

public class Registry {

 private String departmentName;

 // Δημιουργία μιας ArrayList για καταχώρηση των μαθημάτων στην main
 private ArrayList<Course> courses = new ArrayList<>();

 // Κατασκευαστής
 public Registry(String aName) {
  departmentName = aName;
 }

 // Μέθοδος προσθήκης μαθήματος στην ArrayList
 public void addCourse(Course aCourse){

  courses.add(aCourse);

 }

 // Εκτύπωση όλων των πληροφοριών της γραμματείας
 public void printDepartmentData(){

  System.out.println("Department Name: " + departmentName);
  System.out.println("Has the following courses: ");

  for(int i=0 ; i<courses.size(); i++){
   Course c = courses.get(i);
   c.printCourseDetails();
  }
 }

}
```

```
public class Main {

 public static void main(String[] args) {
//  Student s1 = new Student("Nick", 8.4);
//  s1.printInfo();
//
//  GraduateStudent gs1 = new GraduateStudent("Robert", 8.7, "Jones");
//  gs1.printInfo();

  // Αρχή υποκατάστασης ένας πτυχιακός φοιτητής δείχνει σε αντικείμενο μεταπτυχιακού φοιτητή
  Student s1 = new GraduateStudent("Bob", 6.7, "Robers");
  // Πολυμπορφισμός. Θα δείχνει στην printInfo της υποκλάσης. Θα κλειθεί η μέθοδος του αντικειμένου
  s1.printInfo();

 }

}
```

```
public class Student {

 private String name;
 private double gpa;

 public Student(String aName, double aGpa) {
  name = aName;
  gpa = aGpa;
 }

 public void printInfo(){
  System.out.println("Name: " + name);
  System.out.println("GPA: " + gpa);
 }



}
```

```
public class GraduateStudent extends Student {

 private String supervisor;

 public GraduateStudent(String aName, double aGpa, String aSupervisor) {
  super(aName, aGpa);
  supervisor = aSupervisor;
 }

 public void printInfo(){
  super.printInfo();
  System.out.println("Supervisor: " + supervisor);
 }


}
```

```
public class Main {

 public static void main(String[] args) {
  TimeStamp t1 = new TimeStamp(23, 45, 17);

  t1.printTime();

//  t1.hour++;
//  t1.hour++;


  t1.incrementHour();
  t1.incrementHour();

  // Έχοντας public ιδιότητες στην κλάση. Παραβιάζει συνθήκη ώρας 0<= hour < 24. Λεπτά και δευτερόλεπτα μεταξύ του 0 και 60
  t1.printTime();

 }

}
```

```
public class TimeStamp {

 private int hour;
 private int minute;
 private int second;

 public TimeStamp(int h, int m, int s){
  hour =h;
  minute = m;
  second = s;

 }

 public void printTime(){
  System.out.println(hour + ":" + minute + ":" + second);
 }

 // Τηρώ τις αναλοίωτες
 public void incrementHour(){

  hour++;
  if(hour == 24)
   hour = 0;
 }

}
```

```
public class Main {

 public static void main(String[] args) {

  // καλώ μέθοδο χωρίς να έχω δημιουργήσει αντικείμενο
  Soldier.printNumber();

  // Φτιάχνω 2 στρατιωτάκια
  Soldier s1 = new Soldier("XY101");
  Soldier s2 = new Soldier("ZW150");

  // Εκτυπώνω πόσα στρατιωτάκια έχω
  s1.printNumber();
  s2.printNumber();

  // Φτιάχνω ένα ακόμα στρατιωτάκι
  Soldier s3 = new Soldier("XX350");
  s1.printNumber(); // Ρωτάω το πρώτο στρατιωτάκι πόσα έχω δημιουργήσει
  Soldier.printNumber(); // ίδιο αποτέλεσμα με την προηγούμενη

  //Mathematics m1 = new Mathematics();
  System.out.println(Mathematics.powerOfThree(3));
  System.out.println("PI: " + Mathematics.PI); // Επειδή είναι δηλωμένη η ιδιότητα ως public
 }

}
```

```
public class Soldier {

 private String id;
 // στατική ιδιότητα μέσα στην κλάση και όχι για το αντικείμενο
 private static int counter = 0;

 public Soldier(String someId){
  id=someId;
  counter++;
 }

 // στατική μέθοδος για να μπορώ να την καλώ χωρίς να έχω φτιάξει αντικείμενο
 public static void printNumber(){

  System.out.println("Total number: " + counter);
 }


}
```

```
public class Mathematics {

 // final = κάποιος από έξω δεν επιτρέπεται να αλλάξει την τιμή της ιδιότητας
 // static για να μην υπάρχει ανάγκη να γίνει δημιουργία αντικειμένου στην main
 // public για να είναι προσβάσιμη από έξω
 public static final double PI=3.14;

 public static int powerOfThree(int x){

  return x*x*x;
 }

}
```

```
public class Main {

 public static void main(String[] args) {

  BankAccount ba1 = new BankAccount("John", 1500);
// ba1.setName("John");
// ba1.setBalance(1500);
  ba1.printData();

  SavingAccount sa1 = new SavingAccount("Mary", 1800, 0.05);
// sa1.setName("Mary");
// sa1.setBalance(1800);
// sa1.printData();
//
// sa1.setRate(0.05);
// sa1.addInterest();
  sa1.printData();


 }

}
```

```
public class BankAccount{

 // public = ορατή στις υποκλάσεις αλλά όχι στον υπόλοιπο κόσμο. Δεν επιτρέπεται πρόσβαση και αλλαγή τιμής.
 // protected = υποδηλώνει ότι μια ιδιότητα δεν είναι προσπελάσιμη από αντικείμενα άλλων κλάσεων,
 // επιτρέπεται όμως η πρόσβαση από αντικείμενα των υποκλάσεων.

 // εδώ κληρονομείται στην SavingAccount και είναι προσβάσιμη μέσα σε αυτήν
 protected String name;
 protected double balance;

 public BankAccount(String aName, double aBalance){
  name = aName;
  balance = aBalance;
 }

 public void printData(){
  System.out.println("This is the regular bankaccount");
  System.out.println("Name: " + name);
  System.out.println("Balance: " + balance);
 }

 public void deposit(double amount){
  balance += amount;
 }

 public String getName() {
  return name;
 }
 public void setName(String name) {
  this.name = name;
 }
 public double getBalance() {
  return balance;
 }
 public void setBalance(double balance) {
  this.balance = balance;
 }



}
```

```
public class SavingAccount extends BankAccount {

 private double interestRate;


 public SavingAccount(String aName, double aBalance, double aRate){
  // Κλήση κατασκευαστή υπερκλάσης και μεταβίβαση των 2 παραμέτρων
  // ΠΑΝΤΑ καλώ τον κατασκευαστή της υπερκλάσης
  super(aName, aBalance);
  interestRate = aRate;
 }

 public void setRate(double amount){
  interestRate = amount;
 }

 public void addInterest(){
  balance += balance * interestRate;
 }

 // επαναορισμός ή επικάλυψη (override) μεθόδου υπερκλάσης
 public void printData(){
  //κλήση μεθόδου υπερκλάσης
  //super.printData();
  System.out.println("This is a savings bankaccount");
  System.out.println("Name: " + name);
  System.out.println("Balance: " + balance);
  System.out.println("Interest rate: " + interestRate);
 }

}
```

```
public class CheckingAccount extends BankAccount {

 public int transactionsCounter;

 public CheckingAccount(String aName, double aBalance) {
  super(aName, aBalance);
  transactionsCounter = 0;
 }

 public void deposit(double amount){
  // κλήση μεθόδου υπερκλάσης από την υποκλάση
  super.deposit(amount);
  transactionsCounter++;
  if(transactionsCounter >=3)
   balance -=0.5;
 }
}
```

```
import java.util.ArrayList;

import javax.swing.JOptionPane;

public class Main {

 public static void main(String[] args) {

  // 2η εργασία (Λόγω αρχής υποκατάστασης, βάζω στο ArrayList μόνο αναφορές Student)
  ArrayList<Student> students = new ArrayList<>();

  //Εισαγωγή φοιτητή με αλληλεπίδραση με τον χρήστη
  boolean more = true;
  while(more){
   //Μέθοδος που εμφανίζει παράθυρο για εισαγωγή δεδομένων
   //Η μέθοδο είναι static οπότε δεν χρειάστηκε να φτιάξω κλάση.
   String name = JOptionPane.showInputDialog("Enter name: ");
   String id = JOptionPane.showInputDialog("Enter id: ");
   String answer = JOptionPane.showInputDialog("Type of Student: stud, grad, phd ");

   //Αρχή της υποκατάστασης. Δηλώνω μια μεταβλητή, ώστε να βάλω μια student.add στο τέλος
   Student student = null;
   //Τι θα γίνει αν δεν δηλώσει ο χρήστης και δεν μπεί μέσα στις if; Δηλώνω null για αρχικοποίηση.
   //Με else, τότε θα έμπαινε στην τελευταία οπότε δεν θα χρειαζόταν αρχικοποίηση


   //Όταν συγκρίνω αντικείμενα κλάσεων, δεν μπορώ να χρησιμοποιήσω ==. Βλέπει μόνο αν είναι ίσες οι διευθύνσεις. Το ==
MONO σε String
   // Τι θα είναι ίσα στα αντικείμενα; Τα αντικείμενα έχουν πολλές μεταβλητές μέσα
   //equalsIgnoreCase = ανεξάρτητα με το αν είναι κεφαλαία ή μικρά

   //Εισάγω UnderGradStudent την νέα κλάση
   if(answer.equals("stud")){
    String yearText = JOptionPane.showInputDialog("Enter year: ");
    //Μετατρέπω ένα κείμενο που έχει ακέραιο αριθμό και τον μετατρέπει ως μεταβλητή τύπου ακέραιο
    int year = Integer.parseInt(yearText);
    student = new UnderGradStudent(name, id, year);

   }

   if(answer.equals("grad")){
    String supervisor = JOptionPane.showInputDialog("Supervisor: ");
    student = new GraduateStudent(name, id, supervisor);

   }

   if(answer.equals("phd")){
    String thesis = JOptionPane.showInputDialog("Thesis: ");
    student = new PhDStudent(name, id, thesis);

   }

   students.add(student);

   String another = JOptionPane.showInputDialog("Another student: yes, no");
   if(another.equals("no")){
    more = false;
   }

  }
```

```
 //students = όνομα δομής δεδομένων που θέλω να τυπώσω.
 //Student = τύπος δεδομενων μέσα στην δομή δεδομένων
 //s = είναι αυθαίρετο όνομα. Είναι σαν το students.get(i)
 for(Student s: students){
  s.printInfo(); //** ΠΟΛΥΜΟΡΦΙΚΗ ΚΛΗΣΗ **//
 }

 for(Student s: students){
  s.printType();
 }


// //Καταχωρώ φοιτητές (δημιουργώ τα αντικείμενα και θέτω την αναφορά της ArrayList στο αντικείμενο φοιτητή)
// //OCP -> Εισάγω κλάσεις αλλά δεν πειράζω το πρόγραμμα της main πχ σε if ή for (ανοικτή κλειστή σχεδίαση)
// students.add(new Student("John", "ics19047"));
// students.add(new GraduateStudent("Mary", "mai20012", "Roberts"));
// students.add(new Student("Babis", "iis20113"));
// students.add(new PhDStudent("Helen", "phd123", "Software Quality"));

// for(int i=0 ; i<students.size(); i++){
//  //ΔΥΝΑΜΙΚΗ ΔΙΑΣΥΝΔΕΣΗ. Ποιά printInfo είναι; --> ΠΟΛΥΜΟΡΦΙΣΜΟΣ (ανάλογα με τι είναι ο αποδέκτης, επιλέγεται το
printInfo του και αποκρίνεται ανάλογα)
//  students.get(i).printInfo();
// }

// //students = όνομα δομής δεδομένων που θέλω να τυπώσω.
// //Student = τύπος δεδομενων μέσα στην δομή δεδομένων
// //student = είναι αυθαίρετο όνομα. Είναι σαν το students.get(i)
// for(Student student: students){
//  student.printInfo();
// }
//

// Student s1 = new Student("John", "ics20132");
//
// //Βάζω έναν GraduateStudent ως Student λόγω αρχής υποκατάστασης
// Student s2 = new GraduateStudent("Mary", "mai20098", "Nikolaou");
//
// // Στατική διασύνδεση με τον printInfo
// s1.printInfo();
//
// // ΔΥΝΑΜΙΚΗ ΔΙΑΣΥΝΔΕΣΗ ή ΚΑΘΥΣΤΕΡΗΜΕΝΗ ΔΙΑΣΥΝΔΕΣΗ. Την ώρα που διαβάζει, δεν ξέρει ποια μέθοδος θα κληθεί,
την ώρα που τρέχει τον κώδικα αποφασίζει ποια μέθοδος θα κληθεί.
// s2.printInfo();
//
// //GraduateStudent gs1 = new GraduateStudent("Mary", "mai20098", "Nikolaou");
// //gs1.printInfo();


 }

}
```

```
//ΑΦΗΡΗΜΕΝΗ (abstract)= δεν μπορώ να χρησιμοποιήσω στην main την κλάση, έστω αντικείμενο Student
public abstract class Student {

 private String name;
 private String id;

 public Student(String name, String id) {
  this.name = name;
  this.id = id;
 }

 public void printInfo(){
  System.out.println("Name: " + name);
  System.out.println("ID: " + id);
 }

 //Εννοια αφαίρεσης
 //Αυτήν δεν θα την χρησιμοποιήσει κανείς. Μέθοδος αφηρημένη (abstract). Χωρίς υλοποίηση.
 //Η μέθοδος αυτή πρέπει να είναι σε abstract κλάση.
 //Την έχω φτιάξει προς όφελος των υποκλάσεων για να επιτρέψω ΠΟΛΥΜΟΡΦΙΣΜΟ. Να μπορεί να εκτυπώσει η υποκλάση
 public abstract void printType();


}
```

```
// Θέλω στην UnderGradStudent να προσθέσω μια ιδιότητα year. Αν την γράψω στην Student θα κληρονομηθεί στην UnderGradStudent και PhDStudent
//που δεν το θέλω. Οπότε φτιάχνω μια νέα υποκλάση UnderGradStudent

public class UnderGradStudent extends Student {

 private int year;

 public UnderGradStudent(String name, String id, int year) {
  super(name, id);
  this.year = year;
 }

 // Επικάλυψη / επαναορισμός μεθόδου
 public void printInfo(){
  super.printInfo();
  System.out.println("year: " + year);
 }

 //Αρχή υποκατάστασης: Τα αντικείμενα μιας υποκλάσης είναι και αντικείμενο της υπερκλάσης
 public void printType(){
  System.out.println("Hello I am an undergraduate student");
 }

}
```

```
public class PhDStudent extends Student {

 private String thesis;

 public PhDStudent(String name, String id, String thesis) {
  super(name, id);
  this.thesis = thesis;
 }

 // Επικάλυψη / επαναορισμός μεθόδου
 public void printInfo(){
  super.printInfo();
  System.out.println("Thesis: " + thesis);
 }

 //Αρχή υποκατάστασης: Τα αντικείμενα μιας υποκλάσης είναι και αντικείμενο της υπερκλάσης
 public void printType(){
  System.out.println("Hello I am a PhD student");
 }
}
```

```
public class GraduateStudent extends Student {

 private String supervisor;

 public GraduateStudent(String name, String id, String supervisor) {
  super(name, id);
  this.supervisor = supervisor;
 }

 // Επικάλυψη / επαναορισμός μεθόδου
 public void printInfo(){
  super.printInfo();
  System.out.println("Supervisor: " + supervisor);

 //Αρχή υποκατάστασης: Τα αντικείμενα μιας υποκλάσης είναι και αντικείμενο της υπερκλάσης
 }

 public void printType(){
  System.out.println("Hello I am a graduate student");
 }

}
```

```
public class Main {

 public static void main(String[] args) {

  // Δημιουργία ενός αντικειμένου dataset
  DataSet ds = new DataSet();

  // Δημιουργία αντικειμένων BankAccount
  BankAccount ba1 = new BankAccount(1200);
  BankAccount ba2 = new BankAccount(2100);
  BankAccount ba3 = new BankAccount(1600);

  // Δημιουργία αντικειμένων Student
  Student s1 = new Student(8.9);
  Student s2 = new Student(6.7);
  Student s3 = new Student(9.4);

  // Προσθήκη αντικειμένων BankAccount στο Dataset
  ds.add(ba1);
  ds.add(ba2);
  ds.add(ba3);

  // Προσθήκη αντικειμένων Student στο Dataset
//  ds.add(s1);
//  ds.add(s2);
//  ds.add(s3);

  // Εκτύπωση του Dataset με την χρήση της διεπαφής
  System.out.println("Average: " + ds.getAverage());
  System.out.println("Maximum: " + ds.getMaximum().getMeasure());
  System.out.println("Minimum: " + ds.getMinimum().getMeasure());

  // Για BankAccount: Αρχικό πρόγραμμα χωρίς την χρήση της διεπαφής
//  System.out.println("Average: " + ds.getAverage());
//  System.out.println("Maximum: " + ds.getMaximum().getBalance());
//  System.out.println("Minimum: " + ds.getMinimum().getBalance());

  // Για Student: Αρχικό πρόγραμμα χωρίς την χρήση της διεπαφής
//  System.out.println("Average: " + ds.getAverage());
//  System.out.println("Maximum: " + ds.getMaximum().getGPA());
//  System.out.println("Minimum: " + ds.getMinimum().getGPA());
 }

}
```

```
public class DataSet{

 // Για ΔΙΕΠΑΦΗ

 private int counter;
 private double sum;

 // Είναι reference προς μια κλάση Measurable
 private Measurable maximum;
 private Measurable minimum;

 public DataSet(){
  //Εναν ακέραιο, double αν δεν τα αρχικοποιήσω, θα έχουν τιμή 0.
  //Οπότε ψιλοάχρηστο. Πιο πολύ για τον προγραμματιστή είναι αυτό
  counter=0;
  sum=0;
  maximum = null;
  minimum = null;
 }

 // Μέθοδος add Measurable
 public void add(Measurable item){

  sum += item.getMeasure();

  if(counter == 0 || item.getMeasure() > maximum.getMeasure())
   // maximum και item είναι αναφορές,
   //οπότε maximum δείχνει Measurable με μεγαλύτερο υπόλοιπο
   maximum = item;

  if(counter == 0 || item.getMeasure() < minimum.getMeasure())
   minimum = item;

  counter++;

 }

 public double getAverage(){
  if(counter == 0)
   return 0;
  return sum/counter;
 }

 public Measurable getMaximum(){
  return maximum;
 }

 public Measurable getMinimum(){
  return minimum;
 }


 // Για Student
 //
 // private int counter;
 // private double sum;
 //
 // // Είναι reference προς μια κλάση Student
 // private Student maximum;
 // private Student minimum;
 //
 // public DataSet(){
 //  //Εναν ακέραιο, double αν δεν τα αρχικοποιήσω, θα έχουν τιμή 0.
 //  //Οπότε ψιλοάχρηστο. Πιο πολύ για τον προγραμματιστή είναι αυτό
 //  counter=0;
 //  sum=0;
```

```
//  maximum = null;
//  minimum = null;
// }
//
// public void add(Student item){
//
//  sum += item.getGPA();
//
//  if(counter == 0 || item.getGPA() > maximum.getMeasure())
//   // maximum και item είναι αναφορές,
//   //οπότε maximum δείχνει στον φοιτητή με μεγαλύτερο υπόλοιπο
//   maximum = item;
//
//  if(counter == 0 || item.getMeasure() < minimum.getMeasure())
//   minimum = item;
//
//  counter++;
//
// }
//
// public double getAverage(){
//  if(counter == 0)
//   return 0;
//  return sum/counter;
// }
//
// public Student getMaximum(){
//  return maximum;
// }
//
// public Student getMinimum(){
//  return minimum;
// }


 // Για το BankAccount

// private int counter;
// private double sum;
// // Είναι reference προς μια κλάση BankAccount
// private BankAccount maximum;
// private BankAccount minimum;
//
// public DataSet(){
//  //Έναν ακαίρεο, double αν δεν τα αρχικοποιήσω, θα έχουν τιμή 0.
//  //Οπότε ψιλοάχρηστο. Πιο πολύ για τον προγραμματιστή είναι αυτό
//  counter=0;
//  sum=0;
//  maximum = null;
//  minimum = null;
// }
//
// public void add(BankAccount item){
//
//  sum += item.getBalance();
//
//  if(counter == 0 || item.getBalance() > maximum.getBalance())
//   // maximum και item είναι αναφορές,
//   //οπότε maximum δείχνει στον τραπεζικό λογαριασμό με μεγαλύτερο υπόλοιπο
//   maximum = item;
//
//  if(counter == 0 || item.getBalance() < minimum.getBalance())
//   minimum = item;
//
//  counter++;
//
// }
```

```
// public double getAverage(){
//  if(counter == 0)
//   return 0;
//  return sum/counter;
// }
//
// public BankAccount getMaximum(){
//  // Επιστρέφει τραπεζικό λογαριασμό
//  return maximum;
// }
//
// public BankAccount getMinimum(){
// // Επιστρέφει τραπεζικό λογαριασμό
//  return minimum;
// }
}
```

```
//Υπερκλάση και για το BankAccount όσο και το Student
//Επειδή φτιάχνω abstract μέθοδο, πρέπει να έχω και abstract κλάση

//OCP ΑΡΧΗ ΑΝΟΙΚΤΗΣ-ΚΛΕΙΣΤΗΣ ΣΧΕΔΙΑΣΗΣ: είναι ανοικτό σε επέκταση κώδικα πχ προσθήκη κλάσης,
//αλλά είναι κλειστό σε τροποποιήσεις στον υπάρχοντα κώδικα πχ της main σε if, for κλπ

//Αρχή αντιστροφής εξαρτήσεων

//public abstract class Measurable_old {

// // Αφηρημένη χωρίς υλοποίηση μέθοδος άρα κάνω αφηρημένη και την κλάση
// public abstract double getMeasure();

//}
```

```
//Διασύνδεση με δηλώσεις αφηρημένων μεθόδων

public interface Measurable {

 double getMeasure();

}
```

```
public class Student implements Measurable{

 private double gpa;

 // Κατασκευαστής
 public Student(double gpa) {
  //Omit call .. για να μην βγάζει το super όταν πατώ alt+shift+s
  this.gpa = gpa;
 }

 //Αλλάζω από getGPA σε getMeasure
 public double getMeasure(){
  return gpa;
 }

}
```

```
public class BankAccount implements Measurable {

 private double balance;

 // Κατασκευαστής
 public BankAccount(double balance) {
  this.balance = balance;
 }

 //Αλλάζω το getBalance σε getMeasure
 public double getMeasure() {
  return balance;
 }



}
```

```
import javax.swing.JFrame;

public class Main {

 public static void main(String[] args) {

 MyFrame mf = new MyFrame();

 // ΣΕΤ εντολών που εκτελούμε κάθε φορά για να φτιάξουμε παράθυρο
 //Ορατό παράθυρο, ανάλογα με το στάδιο της εκτέλεσης
 mf.setVisible(true);

 //Μέγεθος παραθύρου
 mf.setSize(400,400);

 //Ενας τίτλος πάνω στο παράθυρο
 mf.setTitle("My first window στα ελληνικά");

 //Τι θα κάνει το παράθυρο όταν πατήσουμε το X πάνω στο παράθυρο
 mf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

 }

}
```

```
import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class MyFrame extends JFrame{

 // Έχω κάνει ιδιότητα ένα γραφικό συστατικό (το κείμενο)
 //ώστε να είναι ορατό σε όλη την κλάση
 private JTextField textField;
 private JTextField outputTextField = new JTextField(20);

 //Κατασκευαστής κλάσης
 public MyFrame(){

  //Βήμα 1. Δημηουργία υποδοχέα
  //Οτι ξεκινάει από J είναι αντικείμενο της swing
  JPanel panel = new JPanel();
  //Θέτουμε χρώμα background
  panel.setBackground(Color.DARK_GRAY);
  //Προσθήκη εικόνας
  ImageIcon icon = new ImageIcon("xmas_holidays.jpg");
  JLabel label = new JLabel(icon);

  //Βήμα 2. Δημιουργία γραφικών συστατικών
  textField = new JTextField(10);
  JButton button = new JButton("Press Me!");

  //Βήμα 3. Τοποθέτηση γραφικών συστατικών στον υποδοχέα
  panel.add(textField);
  panel.add(button);

  //Προσθήκη για να εμφανίζεται στην γραφική διεπαφή
  panel.add(outputTextField);

  //Προσθήκη εικόνας
  panel.add(label);

  //Βήμα 4. Προσαρμογή υποδοχέα στο παράθυρο
  this.setContentPane(panel);

  //ΕΗ. Βήμα 3. Προσαρμογή υποδοχέα στο παράθυρο
  ButtonListener listener = new ButtonListener();

  //ΕΗ. Βήμα 4. Σύνδεση ακροατή με πηγή συμβάντων
  button.addActionListener(listener);
 }

 // ΕΗ. Βήμα 1. Φτιάχνω μια εσωτερική κλάση. Δημιουργία κλάσης ακροατή
 // Εκτελεί μια ενέργεια με το πάτημα ενός πλήκτρου
 // Την μέθοδο την καλεί η Java. ActionListener είναι μια διασύνδεση (έχει μια μέθοδο)
 // Πρόσθέτω την μέθοδο της ActionListener πατώντας πάνω στο ButtonListener
 class ButtonListener implements ActionListener{

  //ΕΗ. Βήμα 2. Συγγραφή του εκτελούμενου κώδικα
  @Override
  public void actionPerformed(ActionEvent arg0) {
   String userName = textField.getText();
   //System.out.println("Hello my friend " + userName);
   outputTextField.setText("Hello " + userName);
```

```
  }
 }
}
```

```
public class Main {

 public static void main(String[] args) {

  Student s1 = new Student("John", "ics20133");
  // Φτιάχνω ένα νέο αντικείμενο με τα ίδια στοιχεία
  Student s2 = new Student("John", "ics20133");

  // Όταν στέλνω ένα αντικείμενο μέσα σε print επιστρέφει το όνομα αντικειμένου και την διεύθυνσή του.
  System.out.println(s1);
  System.out.println(s2);

  // Σύγκριση των αντικειμένων με ==
//  // Εκτυπώνει την else (the two object are different) γιατί συγκρίνει αντικείμενα.
//  // Οι διευθύνσεις είναι διαφορετικές. Είναι αυτές που συγκρίνονται με ==.
//  // Συγκρίνω τα αντικείμενα ΜΟΝΟ με μέθοδο equals
//  if(s1==s2)
//   System.out.println("The two objects refer to the same student");
//  else
//   System.out.println("The two objects are different");

  // Σύγκριση των αντικειμένων με equals
  if(s1.equals(s2))
   System.out.println("The two objects refer to the same student");
  else
   System.out.println("The two objects are different");



 }

}
```

```
public class Student{

 private String name;
 private String id;

 public Student(String name, String id) {
  this.name = name;
  this.id = id;
 }

 // Επικάλυψη της toString και αξιοποίηση του πολυμορφισμού.
 // Χωρίς αυτό, όταν εκτελείται η print εμφανίζει Student@ee7d9f1 (Αντικείμενο@διεύθυνση)
 // Η υπογραφή πρέπει να είναι ίδια γιατί στην object η μέθοδος toString εμφανίζεται έτσι
 public String toString(){
  return (name + ", " + id);
 }

 //Επικάλυψη της equals
 public boolean equals(Object obj){

  //Ρητή μετατροπή τύπου από Object σε Student
  Student otherStudent = (Student)obj;

  //Συγκρίνω όνομα μου αν ταυτίζεται με το όνομα του otherStudent
  //Όταν κάνω μια ιδιότητα private δεν είναι ορατή σε άλλες κλάσεις. Στην περίπτωσή μου είναι ίδια κλάση οπότε την χρησιμοποιώ
  //Σύγκριση μπορεί να γίνει με == αλλά καλύτερα να γίνει με equals επειδή η String έχει υλοποιημένη την equals
  if(this.name.equals(otherStudent.name) && this.id.equals(otherStudent.id))
   return true;
  else
   return false;

 }


}
```

```
public class Main {

 public static void main(String[] args) {

  // Εκκίνηση του παραθύρου. Δημιουργώ ανώνυμο αντικείμενο
  new StudentFrame();

  // Εναλλακτικά μπορώ να το δηλώσω και ως StudentFrame sf = new StudentFrame();

 }

}
```

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class StudentFrame extends JFrame{

 //2ο βήμα GUI, Δημιουργία Panel
 private JPanel panel = new JPanel();

 //2ο βήμα GUI, Δημιουργία 4 γραφικών συστατικών
 private JTextField nameField = new JTextField("Enter name");
 private JTextField idField = new JTextField("Enter student id");
 private JTextField courseField = new JTextField("Enter course name");
 private JButton enrollButton = new JButton("Create and Enroll");

 //Φτιάχνω μια λίστα με μαθήματα
 private ArrayList<Course> courses = new ArrayList<>();

 public StudentFrame(){

  //Προσθέτω μαθήματα στην λίστα μου
  courses.add(new Course("Java"));
  courses.add(new Course("Databases"));
  courses.add(new Course("Python"));

  //3ο βήμα GUI, προσθέτω τα συστατικά στο panel
  panel.add(nameField);
  panel.add(idField);
  panel.add(courseField);
  panel.add(enrollButton);

  //4ο βήμα GUI, προσθέτω το panel στο παράθυρο
  this.setContentPane(panel);

  //3. ButtonListener: Δημιουργώ ένα αντικείμενο ακροαρή για το πλήκτρο
  ButtonListener listener = new ButtonListener();

  //4. ButtonListener: Προσθέτω την σύνδεση του αντικειμένου με την πηγή των συμβάντων
  enrollButton.addActionListener(listener);

  //1ο βήμα GUI, Δημιουργία γραφικής διασύνδεσης (παράθυρο)
  this.setVisible(true);
  this.setSize(400,400);
  this.setTitle("Create and Enroll Student");
  this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 }

 // 1. ButtonListener: Φτιάχνω μια εσωτερική κλάση ακροατή για τα συμβάντα
 // Την ActionListener την παίρνω από την awt
 class ButtonListener implements ActionListener{


  //2. ButtonListener: Υλοποιώ την μέθοδο
  @Override
  public void actionPerformed(ActionEvent e) {

   //Διαβάζω το όνομα του φοιτητή και την ταυτότητά του
   String name = nameField.getText();
   String id = idField.getText();
```

```
  // Δημιουργία φοιτητή
  Student newStudent = new Student(name, id);

  //Παίρνω και το μάθημα που εισήγαγε ο χρήστης
  String courseName = courseField.getText();

  // Καταχώρηση του φοιτητή στο μάθημα
  // Πρέπει να κάνω αρχικοποίηση του επιλεγμένου μαθήματος
  Course selectedCourse = null;
  // Διατρέχω τα μαθήματα
  for(Course course: courses)
   // Ελέγχω το επιλεγμένο μάθημα είναι
   if(course.getName().equals(courseName))
    selectedCourse = course;

  //Καταχωρεί τον φοιτητή στο επιλεγμένο μάθημα
  selectedCourse.enrollStudent(newStudent);

  //Εκτύπωση φοιτητών
  selectedCourse.printCourceDetails();
 }

}

}
```

```
public class Student {

 private String name;
 private String id;

 // Κατασκευαστής
 public Student(String name, String id) {

  this.name = name;
  this.id = id;
 }

 // Επικάλυψη της toString
 public String toString(){

  return (name + ", " + id);
 }


}
```

```
import java.util.ArrayList;

public class Course {

 private String name;

 // Φτιάχνω μια λίστα με φοιτητές
 private ArrayList<Student> students = new ArrayList<Student>();

 // Κατασκευαστής
 public Course(String name) {
  this.name = name;
 }

 // Μέθοδος προσθήκης φοιτητή στην λίστα με τους φοιτητές
 public void enrollStudent(Student aStudent){
  students.add(aStudent);

 }

 // Μέθοδος εκτύπωσης του μαθήματος
 public void printCourceDetails(){
  System.out.println("--------------------");
  System.out.println("Course name: " +name);
  System.out.println("has the following students: ");
  for(Student student: students)
   System.out.println(student);

 }

 public String getName(){
  return name;
 }

}
```

```
public class Main {

 public static void main(String[] args) {

  //Δημιουργώ ένα παράθυρο
  new GUI();

 }

}
```

```
import java.awt.Color;
import java.awt.Graphics;

import javax.swing.JFrame;
import javax.swing.JPanel;

public class GUI extends JFrame {

 public GUI(){

  //2. Δημιουργία του board
  Board board = new Board();

  //3. Προσθήκη του Board στο παράθυρο
  this.setContentPane(board);


  //1.  Δημιουργία παραθύρου
  this.setVisible(true);
  this.setSize(500,500);
  this.setTitle("ChessBoard");
  this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

 }

 // Φτιάχνω μια κλάση για να επικαλύψω την paintComponent
 class Board extends JPanel{

  // Δηλώνω διαστάσεις
  private static final int ROWS = 8;
  private static final int COLUMNS = 8;

  public void paintComponent(Graphics g){

   //Για να προσθέσω μια ακόμα ιδιότητα στην paintComponent την καλώ με super
   super.paintComponent(g);

   // Φτιάχνω μια μεταβλητή για να κλιμακώνει το μέγεθος της σκακιέρας (responsive)
   int sqSize1 = this.getHeight() / ROWS;
   int sqSize2 = this.getWidth() / COLUMNS;
   int sqSize = sqSize1;
   if(sqSize2 < sqSize)
    sqSize = sqSize2;

   // Ζωγραφίζει ένα κουτάκι από το σημείο 0,0 και να έχει μέγεθος πλάτος 10 και ύψος 10
   //g.drawRect(0, 0, 10, 10);

//   //Ζωγραφίζω την σκακιέρα μου
//   for(int i=0; i<ROWS; i++){
//    for(int j=0; j<COLUMNS; j++){
//     int x,y;
//     x = j*10; // Για να ξεκινάει από το 0,10
//     y = i*10; // Για να ξεκινάει από 0,10
//     g.drawRect(x, y, 10, 10);
//    }
//   }


   //Ζωγραφίζω την σκακιέρα μου
   for(int i=0; i<ROWS; i++){
    for(int j=0; j<COLUMNS; j++){
     int x,y;
     x = j*sqSize; // Για να ξεκινάει από το 0,sqSize
     y = i*sqSize; // Για να ξεκινάει από 0,sqSize
     g.drawRect(x, y, sqSize, sqSize);
```

```
    // Ζωγραφίζω εναλλάξ κόκκινο και μπλε χρώμα ανάλογα αν είναι ζυγό ή μονό το τετραγωνάκι
    if((i+j)%2 ==0){
     g.setColor(Color.RED); // Αλλάζω τα χρώματα
     g.fillRect(x, y, sqSize, sqSize);

    }

    else{
     g.setColor(Color.BLUE);
     g.fillRect(x, y, sqSize, sqSize);
    }
   }
  }
 }
}


}
```

```
import java.util.ArrayList;

public class Main {

 public static void main(String[] args) {

  //Φτιάχνω μια λίστα
  ArrayList<Ship> ships = new ArrayList<>();

  //Φτιάχνω αντικείμενα πλοία
  Ship ship1 = new Ship("Zeus", 2);
  Ship ship2 = new Ship("Cobacabana", 100);
  Ship ship3 = new Ship("Tinos", 220);
  Ship ship4 = new Ship("Hercules", 180);
  Ship ship5 = new Ship("SuperStar", 240);
  Ship ship6 = new Ship("Olympia", 320);

  //Προσθέτω τα πλοία στην λίστα
  ships.add(ship1);
  ships.add(ship2);
  ships.add(ship3);
  ships.add(ship4);
  ships.add(ship5);
  ships.add(ship6);

  // Εκκίνηση παραθύρου και στέλνω το ArrayList ships στη γραφική διασύνδεση μέσω του κατασκευαστή
  new ContainerFrame(ships);


 }

}
```

```
import java.util.ArrayList;

public class Ship {

 private String name;
 private int capacity;
 // Το αμπάρι του πλοίου. Δείχνει σε ένα αντικείμενο container (αρχή υποκατάστασης, λαμβάνει και τους δυο τύπους)
 private ArrayList<Container> containers = new ArrayList<>();


 // Κατασκευαστής
 public Ship(String name, int capacity) {
  this.name = name;
  this.capacity = capacity;
 }

 // Για να πάρω τα ονόματα των πλοίων
 public String getName(){
  return name;
 }

 // Μέθοδος προσθήκης container αν χωράει στο πλοίο
 public void addContainer(Container aContainer){

  if(containers.size() < capacity){
   containers.add(aContainer);
   System.out.println("The container has been loaded");
  }
  else
   System.out.println("Sorry the ship is full");
 }

 // Υπολογισμός συνολικής χρέωσης
 public double calculateTotalCharge(){
  double total = 0;
  for(Container container: containers)
   total += container.calculateCharge();

  return total;

 }


}
```

```
public class Refrigirator extends Container{

 private double power;

 // Κατασκευαστής
 public Refrigirator(String code, String destination, double power) {
  super(code, destination);
  this.power = power;
 }

 // Μέθοδος υπολογισμού χρέωσης (πολυμοορφισμός)
 public double calculateCharge(){

  return 2000 * power;
 }

}
```

```
public abstract class Container {

 private String code;
 private String destination;

 public Container(String code, String destination) {
  this.code = code;
  this.destination = destination;
 }

 //Δημιουργία μεθόδου για να την χρησιμοποιήσω στην Ship ώστε να υπολογίσω την χρέωση του πλοίου.
 //Την φτιάχνω αφηρημένη (abstract) και μετατρέπω και την κλάση αφηρημένη
 public abstract double calculateCharge();


}
```

```
public class Bulk extends Container {

 private double weight;

 // Κατασκευαστής
 public Bulk(String code, String destination, double weight) {
  super(code, destination);
  this.weight = weight;
 }

 // Μέθοδος υπολογισμού χρέωσης (πολυμορφισμός)
 public double calculateCharge(){

  return 10*weight;
 }

}
```

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;

import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class ContainerFrame extends JFrame {

 //2. GUI:  Δημιουργία του panel
 private JPanel panel = new JPanel();

 //2. GUI:  Δημιουργία γραφικών συστατικών
 // Συστατικά τύπου κειμένου
 private JTextField codeField = new JTextField("Enter code");
 private JTextField destinationField = new JTextField("Enter destination");
 private JTextField weightField = new JTextField("Enter weight");
 private JTextField powerField = new JTextField("Enter power");
 // Συστατικό τύπου κουμπί
 private JButton createBulkButton = new JButton("Create Bulk");
 private JButton createRefrField = new JButton("Create Refr");
 // Συστατικό τύπου λίστα
 private JList list = new JList();

 //Δηλώνω ιδιότητα που δεν δείχνει πουθενά. Θέλω να δείχνει σε όλα τα πλοία. Θα το συνδέσω στον κατασκευαστή
 private ArrayList<Ship> allShips;

 //Δομή δεδομένων model για χρήση στην λίστα
 private DefaultListModel model = new DefaultListModel();

 //Μου στέλνει η Main τις "διευθύνσεις" των αντικειμένων της λίστας. Η χρήση της ships μέσα στον κατασκευαστή
 public ContainerFrame(ArrayList<Ship> ships){

  //Ανάθεση στην ιδιότητα που δημιούργησα να δείχνει στην λίστα που πήρα από την Main
  //Γεφύρωσα το χάσμα της γραφικής διασύνδεσης με τα αντικείμενα της main
  allShips = ships;

  // Εισαγωγή κειμένου επιλογών στην λίστα (πρόσθεση όων των πλοίων)
  // Διατρέχω όλα τα πλοία
  for(Ship ship: allShips)
   model.addElement(ship.getName()); // Προσθήκη πλοίου στην δομή δεδομένων (λίστα) model

  // Εισαγωγή του model στο συστατικό τύπου λίστα
  list.setModel(model);

  //3. GUI:  Εισαγωγή γραφικών συστατικών στο panel
  panel.add(codeField);
  panel.add(destinationField);
  panel.add(weightField);
  panel.add(powerField);
  panel.add(createBulkButton);
  panel.add(createRefrField);
  panel.add(list);

  //4. GUI:  Προσαρμογή του panel πάνω στο παράθυρο
  this.setContentPane(panel);

  // Δημιουργώ ακροατή (βήμα 3ο)
  ButtonListener listener = new ButtonListener();

  //Σύνδεση με τον ακροατή συμβάντων (βήμα 4)
  createBulkButton.addActionListener(listener);
```

```
createRefrField.addActionListener(listener);

//1. GUI: Δημιουργία παραθύρου
this.setVisible(true);
this.setSize(200,400);
this.setTitle("Create and Load Containers");
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

}

// Δημιουργία κλάσης Ακροατή (βήμα 1)
class ButtonListener implements ActionListener{

//Μέθοδος που θα εκτελείται όταν πατάμε το πλήκτρο (βήμα 2)
@Override
public void actionPerformed(ActionEvent arg0) {

//Παίρνω όνομα πλοίου και προορισμό
String code = codeField.getText();
String destination = destinationField.getText();

//Παίρνουμε το πλοίο που επέλεξε ο χρήστης
String selectedShipName = (String)list.getSelectedValue(); // Η getSelectedValue επιστρέφει object οπότε το κάνω String

//Πρέπει να βρω το πλοίο που έχει αυτό το όνομα
Ship selectedShip = null;

for(Ship ship: allShips) // Διατρέχω την λίστα των πλοίων που έχω πάρει από την main
 if(ship.getName().equals(selectedShipName))
  selectedShip = ship;

// Δημιουργώ το container για να μην παραπονιέται μετά την if/else
Container container;

// Πρέπει να καταλάβω ποιο πλήκτρο πατήθηκε
if(arg0.getSource().equals(createBulkButton) ){ // Αν πατήθηκε το createBulkButton

 // Διαβάζω δεδομένα
 String weightText = weightField.getText();
 double weight = Double.parseDouble(weightText); // Αλλάζω το string σε double
 //Φτιάχνω το container με τα δεδομένα που εισήγαγε ο χρήστης
 container = new Bulk(code, destination, weight);


}
else {// Αν πατήθηκε το createRefrField
 String powerText = powerField.getText();
 double power = Double.parseDouble(powerText);
 container = new Refrigirator(code,destination,power);

}

// Βρήκα το πλοίο και του προσθέτω το container
selectedShip.addContainer(container); // Πρέπει να δηλώσω το container πριν το if/else
// Εκτυπώνω
System.out.println("Total charge of ship " + selectedShipName + " is: " + selectedShip.calculateTotalCharge());
 }
 }
}
```

```
import java.util.*;

public class Main {


 public static void main(String[] args) {

  // ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ
  // Φτιάχνω μια συνδεδεμένη λίστα αντικειμένων τύπου Integer
  LinkedList<Integer> list = new LinkedList<>();

  // Κανονικά πρέπει να το γράψω έτσι αλλά η Java, αν το γράψω μόνο 5, το μετατρέπει η ίδια σε αντικείμενο
  list.add(new Integer(5));

  //Εδώ μετατρέπει τους αριθμούς σε αντικείμενα Integer
  list.add(7);
  list.add(3);
  list.add(11);


//  for(Integer i: list)
//   System.out.println(i);

  // Επαναλήπτης που θα διασχίσει ακεραίους της δομής δεδομένων (δουλεύει σε όλες τις δομές)
  Iterator<Integer> iter = list.iterator();

  //Όσο ο επαναλήπτης έχει επόμενο στοιχείο
  while(iter.hasNext()){
   //Φέρτο και τύπωσέ το
   System.out.println(iter.next());
  }

  // Αν θέλω να ξανατρέξω την λίστα, πρέπει να τον φρεσκάρω
  // iter = list.iterator();
 }
}
```

```java
import java.util.*;

public class Main {


 public static void main(String[] args) {

  // ΣΥΝΟΛΑ
  HashSet set = new HashSet(); // Το κάνω έτσι για να κάνω type casting όταν θα διατρέξω το σύνολο
//    HashSet<String> set = new HashSet<String>();

  // Προσθέτω ονόματα στο σύνολο
  set.add("John");
  set.add("Mary");
  set.add("Mary");
  set.add("Bob");
  set.add("John");
  set.add("Nick");

  // Επαναλήπτης
  Iterator iter = set.iterator();
  //Iterator<String> iter = set.iterator();

  // Διατρέχω το σύνολο
  while(iter.hasNext()){
   String name = (String)iter.next(); // Επειδή το hashset έχει object, πρέπει να το μετατρέψω σε String
   System.out.println(name);

   //Επίσης δουλεύει και το
   //System.out.println(iter.next().toString());
  }
   // Τα αποτελέσματα ΔΕΝ έχουν διάταξη και ΔΕΝ έχουν τα διπλότυπα.
   // Αν χρειαστεί να βγάλω διπλότυπα, τα ρίχνω σε σύνολο και βγαίνουν
 }
}
```

```
import java.util.*;

public class Main {


 public static void main(String[] args) {

  // ΑΦΙΑΡΕΣΗ ΔΙΠΛΟΤΥΠΩΝ
  ArrayList<String> inputs = new ArrayList();

  // Προσθέτω ονόματα στην λίστα
  inputs.add("John");
  inputs.add("Mary");
  inputs.add("Mary");
  inputs.add("Bob");
  inputs.add("John");
  inputs.add("Nick");

  // Επαναλήπτης
  Iterator<String> iter = inputs.iterator();

  // Διατρέχω την λίστα
  while(iter.hasNext()){
   String name = iter.next();
   System.out.println(name);

  }

  System.out.println("--------------------");

  // Φτιάχνω ένα Collection και αναθέτω ένα HashSet του οποίου βάζω μέσα την λίστα για να αφαιρέσω τα διπλότυπα
  Collection<String> noDups = new HashSet<String>(inputs);

  // Μηδενισμός του Επαναλήπτη. Μπορώ να χρησιμοποιήσω και άλλον θεωτηρικά.
  iter = noDups.iterator();

  // Διατρέχω το Collection. Βλέπω έχουν βγει τα διπλότυπα
  while(iter.hasNext()){
   String name = iter.next();
   System.out.println(name);

  }
 }
}
```

```
import java.util.*;

public class Main {


 public static void main(String[] args) {

  // ΑΦΙΑΡΕΣΗ ΔΙΠΛΟΤΥΠΩΝ
  ArrayList<String> inputs = new ArrayList();

  // Προσθέτω ονόματα στην λίστα
  inputs.add("John");
  inputs.add("Mary");
  inputs.add("Mary");
  inputs.add("Bob");
  inputs.add("John");
  inputs.add("Nick");

  // Επαναλήπτης
  Iterator<String> iter = inputs.iterator();

  // Διατρέχω την λίστα
  while(iter.hasNext()){
   String name = iter.next();
   System.out.println(name);

  }

  System.out.println("--------------------");

  // Φτιάχνω ένα Collection και αναθέτω ένα HashSet του οποίου βάζω μέσα την λίστα για να αφαιρέσω τα διπλότυπα
  Collection<String> noDups = new HashSet<String>(inputs);

  // Μηδενισμός του Επαναλήπτη. Μπορώ να χρησιμοποιήσω και άλλον θεωτηρικά.
  iter = noDups.iterator();

  // Διατρέχω το Collection. Βλέπω έχουν βγει τα διπλότυπα
  while(iter.hasNext()){
   String name = iter.next();
   System.out.println(name);

  }
 }
}
```

```
import java.util.*;

public class Main {


 public static void main(String[] args) {

 // ΠΡΑΞΕΙΣ ΣΥΝΟΛΩΝ
 Collection<String> set1 = new HashSet<String>();

 set1.add("A");
 set1.add("B");
 set1.add("C");
 set1.add("D");

 Collection<String> set2 = new HashSet<String>();

 set2.add("D");
 set2.add("E");
 set2.add("F");

 // Τελεστής υποσύνολο (έλεγχος αν το set2 είναι υποσύνολο στο set1)
 //returns true if s2 is a subset of s1
 if(set1.containsAll(set2))
  System.out.println("set2 is a subset of s1");
 else
  System.out.println("set2 is NOT a subset of s1");

 // Ένωση 2 συνόλων. Χαλάει το set1
 //transform set1 into the union of set1 and set2
 set1.addAll(set2);
 System.out.println("---Union----");
 System.out.println(set1);

 // Τομή 2 συνόλων. Χαλάει το set1
 //transform set1 into the intersection of set1 and set2
 set1.retainAll(set2);
 System.out.println("---InterSection----");
 System.out.println(set1);

 // Διαφορά δυο συνόλων
 //transform set1 into the set difference of set1 and set2
 //the difference of set1\set2 is the set containing
 //all of the elements in set1 but not in set2
 System.out.println("---Set Difference----");
 set1.add("X");
 System.out.println(set1);
 System.out.println(set2);
 set1.removeAll(set2);
 System.out.println(set1);
 }

}
```

```
import java.util.*;

public class Main {


 public static void main(String[] args) {

  // ΔΕΝΤΡΟ (Τα αποτελέσματα θα είναι ταξινομημένα)
  Collection<String> set = new TreeSet<String>();

  set.add("Olga");
  set.add("Nikos");
  set.add("Babis");
  set.add("Takis");

//  for(String name: set)
//   System.out.println(name);

  Iterator<String> iter = set.iterator();

  // Εμφανίζονται ταξινομημένα (προστίθενται σε δυαδικό δέντρο)
  while(iter.hasNext()){
   System.out.println(iter.next());
  }

 }

}
```

```
import java.util.*;

public class Main {

 public static void main(String[] args) {

  Collection<BankAccount> set = new TreeSet<BankAccount>();

  BankAccount BA1 = new BankAccount("001", 1500, "Papadopoulos");
  BankAccount BA2 = new BankAccount("002", 2500, "Nikolaou");
  BankAccount BA3 = new BankAccount("003", 1000, "Petrou");

  set.add(BA1);
  set.add(BA2);
  set.add(BA3);

  for(BankAccount account: set) {

     System.out.println(account.getId() + ", " +
               account.getBalance() + ", " +
               account.getHolderName());
  }
 }
}

//Δημιουργώ Interface. Για να βάλω κάτι μέσα στην TreeSet πρεπεί ΥΠΟΧΡΕΩΤΙΚΑ να υλοποιεί την διασύνδεση Comparable
class BankAccount implements Comparable {
 private String id;
 private double balance;
 private String holderName;

 // Πρέπει να υλοποιήσω την μέθοδο του interface
 @Override
 public int compareTo(Object other) {
  BankAccount otherAccount = (BankAccount)other; // Κάνω casting BankAccount
  if(this.balance < otherAccount.balance)
   return -1;
  else if(this.balance > otherAccount.balance)
   return 1;
  else
   return 0;

  // Αν ηθελα να ταξινομήσω με βάση τον id
  //return id.compareTo(otherAccount.id);
 }

 // Κατασκευαστής
 public BankAccount(String id, double balance, String holderName) {
  this.id = id;
  this.balance = balance;
  this.holderName = holderName;
 }

 public String getId() {
  return id;
 }

 public double getBalance() {
  return balance;
 }

 public String getHolderName() {
  return holderName;
 }
```

```
import java.util.*;

public class Main {

 public static void main(String[] args) {

 AccountNameComparator myComparator = new AccountNameComparator();

 // Η TreeSet δέχεται ως παράμετρο έναν εξωτερικό συγκριτή (ένα αντικείμενο τύπου Comparator)
 // ΜΟΝΟ ένας συγκριτής κάθε φορά. Αλλιώς πρέπει να τον αλλάζω.
 Collection<BankAccount> set = new TreeSet<BankAccount>(myComparator);

 BankAccount BA1 = new BankAccount("001", 1500, "Papadopoulos");
 BankAccount BA2 = new BankAccount("002", 2500, "Nikolaou");
 BankAccount BA3 = new BankAccount("003", 1000, "Petrou");

 set.add(BA1);
 set.add(BA2);
 set.add(BA3);

 for(BankAccount account: set) {

    System.out.println(account.getId() + ", " +
              account.getBalance() + ", " +
              account.getHolderName());
 }
 }
}

class BankAccount {
 private String id;
 private double balance;
 private String holderName;

 public BankAccount(String id, double balance, String holderName) {
  this.id = id;
  this.balance = balance;
  this.holderName = holderName;
 }

 public String getId() {
  return id;
 }

 public double getBalance() {
  return balance;
 }

 public String getHolderName() {
  return holderName;
 }

}

//Εξωτερικός συγκριτής βάση τραπεζικού Κωδικού
class AccountCodeComparator implements Comparator<BankAccount> {

 public int compare(BankAccount account1, BankAccount account2) {
  String code1 = account1.getId();
  String code2 = account2.getId();

  return code1.compareTo(code2);
 }
}
```

```
//Εξωτερική κλάση για σύγκριση με βάση το όνομα κάτοχου λογαριασμού
class AccountNameComparator implements Comparator<BankAccount> {

 public int compare(BankAccount account1, BankAccount account2) {
  String name1 = account1.getHolderName();
  String name2 = account2.getHolderName();

  return name1.compareTo(name2);
 }
}
```

```
import java.util.HashSet;
import java.util.Iterator;


public class Main {

 public static void main(String[] args) {

  BankAccount BA1 = new BankAccount(1500, "001");
  BankAccount BA2 = new BankAccount(1500, "001");

  HashSet<BankAccount> hashSet = new HashSet<BankAccount>();

  hashSet.add(BA1);
  hashSet.add(BA2);

//  for(BankAccount account: hashSet)
//   System.out.println(account.getCode() + ", "
//                + account.getBalance());

  Iterator iter = hashSet.iterator();

  while(iter.hasNext()){
   BankAccount account = (BankAccount)iter.next();
   System.out.println(account.getCode() + ", " + account.getBalance());

  }

 }

}
```

```
public class BankAccount {

 private double balance;
 private String code;

 public BankAccount(double balance, String code) {
  this.balance = balance;
  this.code = code;
 }

 public double getBalance() {
  return balance;
 }

 public String getCode() {
  return code;
 }


 // Όταν φτιάχνω μια κλάση και θέλω να εισάγω τα αντικείμενά της σε μια HashSet, πέρπει να επικαλύψω την hashcode
 // Επικάλυψη της hashCode για να καταλάβει και να βάλει τους λογαριασμούς σε ίδια θέση
 // αφού έχουν ίδιο κωδικό και υπόλοιπο. Άρα πρέπει να εφαρμόσει το hashCode πάνω στο code που είναι ίδιο.
 public int hashCode() {
  return code.hashCode();
 }

 // Πρέπει να επικαλύψω και την equals μαζί με την hashCode.
 public boolean equals(Object other) {
  BankAccount otherAccount = (BankAccount)other;
  if(this.code.equals(otherAccount.code))
   return true;
  else
   return false;
 }
}
```

```
import java.util.*;

public class Main {

 public static void main(String[] args) {

 ArrayList<String> names = new ArrayList<String>();

 names.add("John");
 names.add("Bob");
 names.add("Nick");
 names.add("Mary");
 names.add("Helen");

 System.out.println("----Sorted----");
 Collections.sort(names); // Ταξινομώ την λίστα
 for(String name: names)
  System.out.println(name);

 System.out.println("----Reversed----");
 Collections.reverse(names); // Αντιστρέφω την ταξινόμηση
 for(String name: names)
  System.out.println(name);

 System.out.println("----Shuffled----");
 Collections.shuffle(names); // Τυχαίο ανακάτεμα. Τυχαία σειρά
 for(String name: names)
  System.out.println(name);

 System.out.println("----Swapped----");
 Collections.swap(names, 2, 3); // Εναλλαγή σειράς
 for(String name: names)
  System.out.println(name);

 System.out.println("----Frequency----");
 names.add("Mary");
 names.add("Mary");
 int freq = Collections.frequency(names, "Mary"); // Εύρεση συχνότητας εμφάνισης
 System.out.println("Frequency of Mary is: " + freq);

 System.out.println("----Min and Max elements----"); // Βρίσκω το ελάχιστο και το μέγιστο
 String max = Collections.max(names);
 String min = Collections.min(names);

 System.out.println("Max is: " + max);
 System.out.println("Min is: " + min);
 }

}
```

```
import java.util.*;

public class Main {

 public static void main(String[] args) {

 ArrayList<String> names = new ArrayList<String>();

 names.add("John");
 names.add("Bob");
 names.add("Nick");
 names.add("Mary");
 names.add("Helen");

 System.out.println("----Sorted----");
 Collections.sort(names); // Ταξινομώ την λίστα
 for(String name: names)
  System.out.println(name);

 System.out.println("----Reversed----");
 Collections.reverse(names); // Αντιστρέφω την ταξινόμηση
 for(String name: names)
  System.out.println(name);

 System.out.println("----Shuffled----");
 Collections.shuffle(names); // Τυχαίο ανακάτεμα. Τυχαία σειρά
 for(String name: names)
  System.out.println(name);

 System.out.println("----Swapped----");
 Collections.swap(names, 2, 3); // Εναλλαγή σειράς
 for(String name: names)
  System.out.println(name);

 System.out.println("----Frequency----");
 names.add("Mary");
 names.add("Mary");
 int freq = Collections.frequency(names, "Mary"); // Εύρεση συχνότητας εμφάνισης
 System.out.println("Frequency of Mary is: " + freq);

 System.out.println("----Min and Max elements----"); // Βρίσκω το ελάχιστο και το μέγιστο
 String max = Collections.max(names);
 String min = Collections.min(names);

 System.out.println("Max is: " + max);
 System.out.println("Min is: " + min);
 }

}
```

```
public class Main {

 public static void main(String[] args) {
  new GUI();

 }

}
```

```
import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Collections;

import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextField;

public class GUI extends JFrame{

 private JPanel panel = new JPanel();
 private JTextField input = new JTextField("add a name here");
 private JButton addButton = new JButton("Add");
 private JButton shuffleButton = new JButton("Shuffle");

 // Δημιουργία λίστας
 private JList list = new JList();

 // Τροφοδοτεί την λίστα
 private DefaultListModel<String> model = new DefaultListModel<>();

 // Προσθήκη μπάρας κύλισης
 private JScrollPane scrollPane;

 public GUI(){

  list.setModel(model); // Προσθέτω την δομή πάνω στην λίστα
  scrollPane = new JScrollPane(list); // Περιτυλίγει το γραφικό συστατικό list

  // Προσθέτω τα συστατικά στο panel
  panel.add(input);
  panel.add(addButton);
  panel.add(scrollPane); // Βάζω την λίστα+την μπάρα κύλισης στο panel
  panel.add(shuffleButton);

  // Προσθέτω το panel στο παράθυρο
  this.setContentPane(panel);
  panel.setBackground(Color.GREEN); //Χρώμα παραθύρου

  // Σύμπτυξη όλων των ενεργειών του Button
  // Δημιουργία ButtonListener
  // Συγγραφή actionPerformed
  // Κατασκευή listener
  // Σύνδεση listener με button
  addButton.addActionListener(new ActionListener () {

   @Override
   public void actionPerformed(ActionEvent arg0) {

    model.addElement(input.getText()); // Παίρνω ότι έχει πληκτρολογήσει ο χρήστης και το βάζει στην λίστα

   }


  });

  // Ανώμνυμη κλάση ActionListener και πρέπει να φτιάξω την actionPerformed
  shuffleButton.addActionListener(new ActionListener() {
```

```
    @Override
    public void actionPerformed(ActionEvent arg0) {

    //Θέλω να ταξινομίσω την δομή λίστα (models) που δεν είναι τύπου λίστας
    //Η μέθοδος elements επιστρέφει ένα enumeration interface και η .list μου κάνει το μοντέλο τύπο λίστας
    //Πάντρεμα δομής σε μια άλλη
    ArrayList<String> namesList = Collections.list(model.elements());

    Collections.shuffle(namesList); // Αφού έγινε λίστα, μπορώ να εφαρμόσω μια μέθοδο shuffle

    model.clear(); //καθαρισμός μοντέλου

    // Ξαναβάζω τα στοιχεία ανακατεμένα
    for(String name: namesList)
     model.addElement(name);

    }


  });

  this.setVisible(true);
  this.setSize(400,400);
  this.setTitle("Algorithms");
  this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 }

}
```

```
import java.util.HashMap;
import java.util.Iterator;


public class Main {

 public static void main(String[] args) {

 // Δομή δεδομένων που κάνει αντιστοιχία σε 2 στοιχεία (key --> value)
 // Τα κλειδιά πρέπει να είναι μοναδικά. Είναι Set οπότε ΔΕΝ πρέπει να είναι ίδια
 // ΑΝ ΖΗΤΑΕΙ ΑΝΤΟΙΣΤΟΙΧΙΣΗ ΤΙΜΩ, ΦΩΝΑΖΕΙ ΝΑ ΕΙΝΑΙ MAP
 HashMap<String, String> map = new HashMap<String, String>();

 map.put("Mary", "2310-538759");
 map.put("Nick", "2310-766350");
 map.put("Helen", "2310-845788");
 map.put("Mike", "2310-224990");

 // Πως διατρέχω δομή τύπου map για να δω τα στοιχεία της
 for(String key: map.keySet()){ //keySet επιστρέφει το σύνολο των κλειδιών
  String value = map.get(key); // Παίρνω την τιμή που αντιστοιχεί στο key
  System.out.println(key + ": " + value);
 }

 // Εκτύπωση με Iterator
 System.out.println("-----------------");
 Iterator<String> i = map.keySet().iterator(); //keySet επιστρέφει το σύνολο των κλειδιών
 while(i.hasNext()) {
  String key = i.next();
  String value = (String)map.get(key); // παίρνουμε την τιμή που αντιστοιχεί στο κλειδί

  System.out.println("Key: " + key + " Value: " + value);
 }

 map.remove("Helen"); // Αφαίρεση της εγγραφής Helen
 System.out.println("------------");
 for(String key: map.keySet()){ //keySet επιστρέφει το σύνολο των κλειδιών
  String value = map.get(key); // Παίρνω την τιμή που αντιστοιχεί στο key
  System.out.println(key + ": " + value);
 }

 }

}
```

```
import java.util.ArrayList;
import java.util.HashMap;

public class Main {

 public static void main(String[] args) {

  // Δομή map με όνομα και λίστα με βιβλία
  HashMap<String, ArrayList<Book>> library = new HashMap<>();

  // Δημιουργία λίστας βιβλίων του John
  ArrayList<Book> johnsBooks = new ArrayList<>();

  johnsBooks.add(new Book("The Alchemist"));
  johnsBooks.add(new Book("The chatcher in the Rye"));

  // Δημιουργία λίστας βιβλίων της Mary
  ArrayList<Book> marysBooks = new ArrayList<>();

  marysBooks.add(new Book("Ti mas kryvoyn"));
  marysBooks.add(new Book("Giati mas psekazoun?"));

  // Προσθήκη των λιστών στο hashmap και αντοιστοιχία των κλειδιών
  library.put("John", johnsBooks);
  library.put("Mary", marysBooks);

  // Εκτύπωση
  for(String member : library.keySet()){
   System.out.println(member + " has borrowed the following books");
   ArrayList<Book> books = library.get(member);
   for(Book book: books)
    System.out.println(book.getTitle());
   System.out.println("----------------");

  }


 }
}

class Book {

 private String title;

 public Book(String title) {

  this.title = title;
 }

 public String getTitle(){

  return title;
 }

}
```

```
public class Main {

 /**
  * @param args
  */
 public static void main(String[] args) {
  new MyFrame();

 }

}
```

```
import javax.swing.JFrame;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;


public class MyFrame extends JFrame {

 public MyFrame() {

  DefaultCategoryDataset dataset = new DefaultCategoryDataset();// Κλάση της βιβλιοθήκης που βάλαμε
  // Προσθέτω τιμές (το Classes είναι η σειρά δεδομένων που ανήκουν. Μπορεί πχ να έχω και άλλη "γραμμή")
  dataset.addValue(212, "Classes", "JDK 1.0");
  dataset.addValue(504, "Classes", "JDK 1.1");
  dataset.addValue(1520, "Classes", "JDK 1.2");
  dataset.addValue(1842, "Classes", "JDK 1.3");
  dataset.addValue(2991, "Classes", "JDK 1.4");

  // Μέθοδος για να παράγω το διάγραμμα
  //JFreeChart chart = ChartFactory.createLineChart("Java Evolution", "Release", "Number of classes", dataset); // Διάγραμμα
Γραμμή
  //JFreeChart chart = ChartFactory.createBarChart("Java Evolution", "Release", "Number of classes", dataset); // Διάγραμμα
μπάρας
  //JFreeChart chart = ChartFactory.createBarChart3D("Java Evolution", "Release", "Number of classes", dataset); // Διάγραμμα
μπάρας 3d

  //Γυρνάω οριζόντια και βάζω το true για να μου δείχνει δεδομένα όταν αφήνω πάνω σε μια μπάρα το ποντίκι
  JFreeChart chart = ChartFactory.createBarChart3D("Java Evolution", "Release", "Number of classes", dataset,
PlotOrientation.HORIZONTAL, false, true, false);

  // Φτιάχνω το panel ως ChartPanel
  ChartPanel chartPanel = new ChartPanel(chart);

  // Προσθήκη του panel στο παράθυρο
  this.setContentPane(chartPanel);

  // Παράθυρο
  this.setVisible(true);
  this.setSize(400, 400);
  this.setTitle("Charts");
  this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

 }


}
```

```
public class Main {

 public static void main(String[] args) {


  GUI gui = new GUI();

 }

}
```

```
import java.awt.Color;
import java.awt.Graphics;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;

import javax.swing.JFrame;
import javax.swing.JPanel;

public class GUI extends JFrame {

 private Board board;

 public GUI() {

 board = new Board();
 this.setContentPane(board);

 //Ακροατής σε συμβάντα τύπου 'κλικ' στο ποντίκι
 //MouseClickListener listener = new MouseClickListener();
 //board.addMouseListener(listener);

 //Ακροατής σε συμβάντα τύπου 'drag & drop' στο ποντίκι
 MouseMoveListener listener = new MouseMoveListener();
 board.addMouseMotionListener(listener);

 this.setVisible(true);
 this.setSize(400, 400);
}

//Χειρισμός συμβάντων κλικ ποντικιού
// Βιβλιοθήκη MouseListener
// Εσωτερική της GUI για να μπορώ να βλέπω το board
class MouseClickListener implements MouseListener {

 // Αφηρημένες μεθόδους

 @Override
 public void mouseClicked(MouseEvent event) {

  // Παίρνω συντεταγμένες που ο χρήστης έκανε κλικ με το ποντίκι
  int x = event.getX();
  int y = event.getY();

  board.setXYCoordinates(x, y); // Θέτει τις συντεταγμένες στην σκακιέρα
  board.repaint(); //Ξανασχεδιάζει τον εαυτό της
 }

 @Override
 public void mouseEntered(MouseEvent arg0) {
  // TODO Auto-generated method stub

 }

 @Override
 public void mouseExited(MouseEvent arg0) {
  // TODO Auto-generated method stub

 }

 @Override
 public void mousePressed(MouseEvent arg0) {
  // TODO Auto-generated method stub

 }
```

```
 @Override
 public void mouseReleased(MouseEvent arg0) {
  // TODO Auto-generated method stub

 }

}

class MouseMoveListener implements MouseMotionListener {

 // Αφηριμένες μεθόδους

 @Override
 public void mouseDragged(MouseEvent event) {

  // Παίρνω συντεταγμένες που ο χρήστης έκανε κλικ με το ποντίκι
  int x = event.getX();
  int y = event.getY();

  board.setXYCoordinates(x, y); // Θέτει τις συντεταγμένες στην σκακιέρα
  board.repaint(); //Ξανασχεδιάζει τον εαυτό της

 }

 @Override
 public void mouseMoved(MouseEvent arg0) {
  // TODO Auto-generated method stub

 }

}


}

class Board extends JPanel {

 // Συντεταγμένες που θα βγαίνει στο πιόνι
 private int xCoord = 0;
 private int yCoord = 0;

 // Θέτει τις τιμές που θα σχεδιαστεί ο δίσκος (μέθοδος set)
 //μέθοδος που θέτει συντεταγμένες στο 'πιόνι' που θα σχεδιαστεί
 public void setXYCoordinates(int x, int y) {
  xCoord = x;
  yCoord = y;
 }

 //Επικάλυψη του painComponent και ζωγραφίζουμε σκακιέρα
 public void paintComponent(Graphics g) {
  super.paintComponent(g);
  int sqSize1 = this.getWidth() / 8;
  int sqSize2 = this.getHeight() / 8;

  int sqSize = sqSize1;
  if(sqSize2 < sqSize1)
   sqSize = sqSize2;

  for(int i=0; i<8; i++) {

   for(int j=0; j<8; j++) {
    int x = j * sqSize;
    int y = i * sqSize;
    //g.drawRect(x, y, sqSize, sqSize);
    if( (i+j)%2 != 0) {
     g.setColor(Color.BLUE);
```

```
      g.fillRect(x, y, sqSize, sqSize);
    }
    else {
      g.setColor(Color.RED);
      g.fillRect(x, y, sqSize, sqSize);
    }
   }

  }

  //Σχεδίαση πιονιού. Αλλάζω το χρώμα της γραφίδας
  //προκειται για οβάλ (κύκλος) που σχεδιάζεται εντός ορθογωνίου (τετραγώνου)
  g.setColor(Color.GREEN);

  // Ζωγραφίζω ένα οβάλ x,y εκεί που ξεκινάει και έχει μέγεθος ίσο με το μέγεθος του τετραγωνιδίου
  // (sqSize/2 για να μην βγαίνει από σκακιέρα).
  // η μετατόπιση κατά sqSize/2 γινεται ώστε το κλίκ να αντιστοιχεί στο κέντρο του κύκλου
  g.fillOval(xCoord-sqSize/2, yCoord-sqSize/2, sqSize, sqSize);

 }



}
```

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;

public class Main {

 public static void main(String[] args) {

  // Θα προσθέτουμε στοιχεία σε ascii αρχείο.
  ArrayList<String> names = new ArrayList<>();

  names.add("John");
  names.add("Mary");
  names.add("Helen");
  names.add("Bob");
  names.add("Nick");

  // Δημιουργώ ένα αρχείο
  File f = new File("names.txt");
  //Δημιουργώ ένα αντικείμενο FileWriter για να γράψω σε αρχείο και στέλνω το αρχείο που δημιούργησα πιο πάνω.
  // Στην αρχή, μας εμφανίζει ένα checked exception.
  // Πρέπει να κάνω χειρισμό. Επιλέγω try/catch
  try {
   FileWriter writer = new FileWriter(f);

   for(String name: names){
    writer.write(name); // Εγγραφή στο αρχείο
    writer.write(System.lineSeparator()); // Αλλαγή γραμμής στο σύστημα που εκτελείται ο κώδικας
   }

   writer.close(); // Κλείνω το ρεύμα...

  } catch (IOException e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  }

 }

}
```

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;

public class Main {

 public static void main(String[] args) {

  // Θα προσθέτουμε στοιχεία σε αρχείο κειμένου.
  ArrayList<String> names = new ArrayList<>();

  names.add("John");
  names.add("Mary");
  names.add("Helen");
  names.add("Bob");
  names.add("Nick");

  // Δημιουργώ ένα αρχείο
  File f = new File("names.txt");
  //Δημιουργώ ένα αντικείμενο FileWriter για να γράψω σε αρχείο και στέλνω το αρχείο που δημιούργησα πιο πάνω.
  // Στην αρχή, μας εμφανίζει ένα checked exception.
  // Πρέπει να κάνω χειρισμό. Επιλέγω try/catch
  try {
   FileWriter writer = new FileWriter(f);

   for(String name: names){
    writer.write(name); // Εγγραφή στο αρχείο
    writer.write(System.lineSeparator()); // Αλλαγή γραμμής στο σύστημα που εκτελείται ο κώδικας
   }

   writer.close(); // Κλείνω το ρεύμα...

  } catch (IOException e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  }

  new GUI();

 }

}
```

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class GUI extends JFrame {

 private JPanel panel = new JPanel(); // Πάνελ
 private JButton openFileButton = new JButton("Open File"); // Κουμπί
 private JFileChooser fc = new JFileChooser(); // Γραφικό συστατικό επιλογής αρχείου

 public GUI(){

  panel.add(openFileButton);
  this.setContentPane(panel);

  //Δημιουργώ ανώνυμη κλάση και το interface που την υλοποιεί...
  openFileButton.addActionListener(new ActionListener(){

   @Override
   public void actionPerformed(ActionEvent arg0) {
    //Ανοίγει το παράθυρο του FileChooser
    //Μητρικό παράθυρο panel
    //Επιστρεφόμενη ακέραια τιμή στο returnValue
    int returnValue = fc.showOpenDialog(panel);

    // Αν επέλεξε το open, τότε επέλεξα να ανοίξει το αρχείο
    if(returnValue == JFileChooser.OPEN_DIALOG){
     //κώδικας για ανάγνωση αρχείο κειμένου

     // Λήψη αρχείου
     File file = fc.getSelectedFile();

     //Περιτυλίγει το FileReader που περιτυλίγει το file
     //Αποδοτικό. Παίρνει μεγάλα κομμάτια από το αρχείο και η προσπέλαση στο δίσκο γίνεται πιο αραιά (γρήγορος)
     // Παράγεται εξαίρεση (διότι έχουμε αρχείο)
     try {
      BufferedReader reader = new BufferedReader(new FileReader(file));

      //Διαβάζω και μπορεί να εμφανίσει εξαίρεση. Προσθέτω ένα catch ακόμα
      String line = reader.readLine(); // Διαβάζει γραμμή ολόκληρη...
      while(line!=null){
       System.out.println(line);
       line = reader.readLine();
      }

      reader.close(); // κλείνω το ρεύμα

     } catch (FileNotFoundException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
     } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
     }

    }
```

```
    }


  });


  this.setVisible(true);
  this.setSize(300, 300);
  this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 }

}
```

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;

public class Main {

 public static void main(String[] args) {

  Employee e = new Employee("John");

  // Δημιουργία αρχείου για να αποθηκεύσω τον υπάλληλο (ser = serialization)
  File file = new File("employee.ser");

  //Εμφάνιση exceptions
  FileOutputStream fileOut;
  try {
   fileOut = new FileOutputStream(file);

   //Υψηλότερου επιπέδου εντολές. Γράφω ένα αντικείμενο καλώντας εντολή writeobject
   ObjectOutputStream out = new ObjectOutputStream(fileOut);

   out.writeObject(e); // Γράφω τον υπάλληλο στο αρχείο
   System.out.println("Object has been serialized");
   // Κλείνω ρεύματα με αντίστροφη σειρά
   out.close();
   fileOut.close();

  } catch (FileNotFoundException e1) {
   // TODO Auto-generated catch block
   e1.printStackTrace();
  } catch (IOException e1) {
   // TODO Auto-generated catch block
   e1.printStackTrace();
  }

 }

}
```

```
import java.io.Serializable;

// Πρέπει να κάνω υλοποίηση της διασύνδεσης για να μπορούν να πάνε σε δυαδικό αρχείο (βγάζει σφάλμα αν δεν έχει)
public class Employee implements Serializable{

 private String name;

 public Employee(String name){
  this.name = name;
 }

 public String getName(){
  return name;
 }

}
```

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.ObjectInputStream;

public class Main {

 public static void main(String[] args) {

  // Θα διαβάσουμε ένα αντικείμενο από ένα αρχείο, οπότε πρέπει να το δημιουργήσω αρχικά.
  Employee e = null;

  // Ανάγνωση αντικειμένου υπάλληλος από αρχείο
  // Εμφανίζει exception οπότε try/catch
  File file = new File("employee.ser"); // Αναπαράσταση του αρχείου δίσκου
  try {
   FileInputStream inputStream = new FileInputStream(file); // Το στέλνω για να το διαβάσει επειδή δεν μπορώ να το διαβάσω κατευθείαν

   ObjectInputStream in = new ObjectInputStream(inputStream); // Επειδή δεν μπορώ να το διαβάσω byte προς byte ο στέλνω για να διαβάσει ολόκληρα αντικείμενα

   // Αναθέτω το αντικείμενο που διάβασα σε ένα αντικείμενο e (ΡΗΤΗ ΜΕΤΑΤΡΟΠΗ ΕΔΩ γιατί θεωρητικά δεν ξέρω τι επιστρέφει)
   // Εμφανίζει exception (δεν μπορεί να καταλάβει την κλάση)
   e = (Employee)in.readObject();

   System.out.println("Object has been deserialized");
   // Εκτυπώνω το όνομα
   System.out.println("The retrieved employee object has name: " + e.getName());
   // Κλείνω ρεύματα
   in.close();
   inputStream.close();

  } catch (FileNotFoundException e1) {
   // TODO Auto-generated catch block
   e1.printStackTrace();
  } catch (IOException e1) {
   // TODO Auto-generated catch block
   e1.printStackTrace();
  } catch (ClassNotFoundException e1) {
   // TODO Auto-generated catch block
   e1.printStackTrace();
  }


 }

}
```

```
import java.io.Serializable;

// Πρέπει να κάνω υλοποίηση της διασύνδεσης για να μπορούν να πάνε σε δυαδικό αρχείο (βγάζει σφάλμα αν δεν έχει)
public class Employee implements Serializable{

 private String name;

 public Employee(String name){
  this.name = name;
 }

 public String getName(){
  return name;
 }

}
```

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.util.ArrayList;

public class Main {

 public static void main(String[] args) {

  // Θέλω να αποθηκεύσω μια λίστα υπαλλήλων σε ένα δυαδικό αρχείο
  Employee e1 = new Employee("John");
  Employee e2 = new Employee("Bob");
  Employee e3 = new Employee("Mary");

  ArrayList<Employee> employees = new ArrayList<>();
  employees.add(e1);
  employees.add(e2);
  employees.add(e3);

  File file = new File("Employees.ser"); // Φτιάχνουμε ένα δυαδικό αρχείο

  // Θα γράψουμε στο δυαδικό αρχείο, όμως θα μας στείλει exception όπότε μπαίνει σε try/catch
  try {
   FileOutputStream outputStream = new FileOutputStream(file);
   ObjectOutputStream out = new ObjectOutputStream(outputStream);

   out.writeObject(employees); // ΕΔΩ ΑΠΟΘΗΚΕΥΟΥΜΕ ΟΛΟΚΛΗΡΗ ΤΗΝ ARRAYLIST

   System.out.println("Employees have been serialized");

   out.close();
   outputStream.close();

  } catch (FileNotFoundException e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  } catch (IOException e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  }

 }

}
```

```
import java.io.Serializable;

// Πρέπει να κάνω υλοποίηση της διασύνδεσης για να μπορούν να πάνε σε δυαδικό αρχείο (βγάζει σφάλμα αν δεν έχει)
public class Employee implements Serializable{

 private String name;

 public Employee(String name){
  this.name = name;
 }

 public String getName(){
  return name;
 }

}
```

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.util.ArrayList;

public class Main {

 public static void main(String[] args) {

  // Φτιάχνω μια κενή λίστα για να μπορεί να δεχτεί την ArrayList που θα διαβάσει από το αρχείο
  ArrayList<Employee> employees = null;

  // Ανάγνωση αντικειμένου υπάλληλος από αρχείο
  // Εμφανίζει exception οπότε try/catch
  File file = new File("Employees.ser"); // Αναπαράσταση του αρχείου δίσκου
  try {
   FileInputStream inputStream = new FileInputStream(file); // Το στέλνω για να το διαβάσει επειδή δεν μπορώ να το διαβάσω κατευθείαν

   ObjectInputStream in = new ObjectInputStream(inputStream); // Επειδή δεν μπορώ να το διαβάσω byte προς byte ο στέλνω για να διαβάσει ολόκληρα αντικείμενα

   // Αναθέτω το αντικείμενο που διάβασα σε ένα αντικείμενο employees (ΡΗΤΗ ΜΕΤΑΤΡΟΠΗ ΕΔΩ γιατί θεωρητικά δεν ξέρω τι επιστρέφει)
   // Εμφανίζει exception η readObject (δεν μπορεί να καταλάβει την κλάση)
   employees = (ArrayList<Employee>)in.readObject();

   System.out.println("Object has been deserialized");
   // Εκτυπώνω το όνομα
   for(Employee e: employees)
    System.out.println("The retrieved employee object has name: " + e.getName());
   // Κλείνω ρεύματα
   in.close();
   inputStream.close();

  } catch (FileNotFoundException e1) {
   // TODO Auto-generated catch block
   e1.printStackTrace();
  } catch (IOException e1) {
   // TODO Auto-generated catch block
   e1.printStackTrace();
  } catch (ClassNotFoundException e1) {
   // TODO Auto-generated catch block
   e1.printStackTrace();
  }


 }

}
```

```
import java.io.Serializable;

// Πρέπει να κάνω υλοποίηση της διασύνδεσης για να μπορούν να πάνε σε δυαδικό αρχείο (βγάζει σφάλμα αν δεν έχει)
public class Employee implements Serializable{

 private String name;

 public Employee(String name){
  this.name = name;
 }

 public String getName(){
  return name;
 }

}
```

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.util.ArrayList;

public class Main {

 public static void main(String[] args) {

  // Θέλω να αποθηκεύσω μια λίστα υπαλλήλων με δικό του αμάξι σε ένα δυαδικό αρχείο

  Car car1 = new Car("Fiat", 1000);
  Car car2 = new Car("Mazda", 2500);
  Car car3 = new Car("Toyota", 1200);

  Employee e1 = new Employee("John", car2);
  Employee e2 = new Employee("Bob", car1);
  Employee e3 = new Employee("Mary", car3);

  ArrayList<Employee> employees = new ArrayList<>();
  employees.add(e1);
  employees.add(e2);
  employees.add(e3);

  File file = new File("Employees.ser");

  try {
   FileOutputStream outputStream = new FileOutputStream(file);
   ObjectOutputStream out = new ObjectOutputStream(outputStream);
   out.writeObject(employees); // ΕΔΩ ΑΠΟΘΗΚΕΥΟΥΜΕ ΟΛΟΚΛΗΡΗ ΤΗΝ ARRAYLIST ΜΑΖΙ ΜΕ ΤΟ ΑΜΑΞΙ ΤΟΥ

   System.out.println("Employees have been serialized");
   // Κλείνουμε τα ρεύματα
   out.close();
   outputStream.close();

  } catch (FileNotFoundException e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  } catch (IOException e) {
   // TODO Auto-generated catch block
   e.printStackTrace();
  }


 }

}
```

```
import java.io.Serializable;

// Πρέπει να κάνω υλοποίηση της διασύνδεσης για να μπορούν να πάνε σε δυαδικό αρχείο (βγάζει σφάλμα αν δεν έχει)
public class Employee implements Serializable{

 private String name;
 private Car ownedCar;

 public Employee(String name, Car aCar){
  this.name = name;
  this.ownedCar = aCar;
 }

 public String getName(){
  return name;
 }

 public Car getCar(){
  return ownedCar;
 }

}
```

```java
import java.io.Serializable;

// Για αποθήκευση σε δυαδικό αρχείο
public class Car implements Serializable{

 private String brand;
 private int cc;

 public Car(String brand, int cc) {

  this.brand = brand;
  this.cc = cc;
 }

 public String getDetails(){
  return (brand + ", " + cc);
 }


}
```

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.util.ArrayList;

public class Main {

 public static void main(String[] args) {

  // Φτιάψνω μια κενή λίστα που θα αποθηκεύσω την λίστα που θα διαβάσω από το αρχείο
  ArrayList<Employee> employees = null;

  // Ανάγνωση αντικειμένου υπάλληλος από αρχείο
  // Εμφανίζει exception οπότε try/catch
  File file = new File("Employees.ser"); // Αναπαράσταση του αρχείου δίσκου
  try {
   FileInputStream inputStream = new FileInputStream(file); // Το στέλνω για να το διαβάσει επειδή δεν μπορώ να το διαβάσω κατευθείαν

   ObjectInputStream in = new ObjectInputStream(inputStream); // Επειδή δεν μπορώ να το διαβάσω byte προς byte ο στέλνω για να διαβάσει ολόκληρα αντικείμενα

   // Αναθέτω το αντικείμενο που διάβασα σε ένα αντικείμενο e (ΡΗΤΗ ΜΕΤΑΤΡΟΠΗ ΕΔΩ γιατί θεωρητικά δεν ξέρω τι επιστρέφει)
   // Εμφανίζει exception (δεν μπορεί να καταλάβει την κλάση)
   employees = (ArrayList<Employee>)in.readObject();

   System.out.println("Object has been deserialized");
   // Εκτυπώνω το όνομα
   for(Employee e: employees){
    System.out.println("The retrieved employee object has name: " + e.getName());
    System.out.println("has the following car: ");
    System.out.println(e.getCar().getDetails());
   }
   // Κλείνω ρεύματα
   in.close();
   inputStream.close();

  } catch (FileNotFoundException e1) {
   // TODO Auto-generated catch block
   e1.printStackTrace();
  } catch (IOException e1) {
   // TODO Auto-generated catch block
   e1.printStackTrace();
  } catch (ClassNotFoundException e1) {
   // TODO Auto-generated catch block
   e1.printStackTrace();
  }


 }

}
```

```
import java.io.Serializable;

// Πρέπει να κάνω υλοποίηση της διασύνδεσης για να μπορούν να πάνε σε δυαδικό αρχείο (βγάζει σφάλμα αν δεν έχει)
public class Employee implements Serializable{

 private String name;
 private Car ownedCar;

 public Employee(String name, Car aCar){
  this.name = name;
  this.ownedCar = aCar;
 }

 public String getName(){
  return name;
 }

 public Car getCar(){
  return ownedCar;
 }

}
```

```java
import java.io.Serializable;

// Για αποθήκευση σε δυαδικό αρχείο
public class Car implements Serializable{

 private String brand;
 private int cc;

 public Car(String brand, int cc) {

  this.brand = brand;
  this.cc = cc;
 }

 public String getDetails(){
  return (brand + ", " + cc);
 }


}
```