183. Πώς επιτυγχάνουμε τη διάταξη των αποτελεσμάτων του ερωτήματος; Δώστε ένα παράδειγμα όπου θα επιλέγεται από ένα πίνακα το ονοματεπώνυμο ενός σπουδαστή και θα εμφανίζονται τα αποτελέσματα ταξινομημένα κατά αύξουσα αλφαβητική σειρά στο όνομα και κατά φθίνουσα αλφαβητική σειρά στο επώνυμο.

Με την εντολή **ORDER BY** για **αύξουσα** σειρά και την εντολή **ORDER BY DESC** για **φθίνουσα** σειρά.
π.χ. **Αύξουσα αλφαβητική σειρά**
SELECT * FROM Students
ORDER BY Name, Surmame;

Φθίνουσα αλφαβητική σειρά
SELECT * FROM Students
ORDER BY Surmame DESC;

184. Εξηγήστε τον ακόλουθο κώδικα:

```
SELECT NAME
FROM WORKER W WHERE EXISTS
(SELECT * FROM CONNECT C WHERE ID=C.CONNECT_ID AND
W.Name=C.CONNECT_Name);
```

Το παραπάνω query μας επιστρέφει τα ονόματα τον εργατών όπου υπάρχει το id του εργάτη στον πίνακα CONNECT και το όνομα του υπάρχει και στον πίνακα WORKER και στον CONNECT.

185. Δώστε 3 τύπους συνένωσης στην SQL (όχι το απλό JOIN) και για κάθε έναν από αυτούς δώστε ένα παράδειγμα με χρήση κώδικα SQL.

- **INNER JOIN:** Το INNER JOIN επιλέγει όλες τις σειρές και από τους δύο πίνακες, εφόσον υπάρχει αντιστοιχία μεταξύ των στηλών. Εάν υπάρχουν εγγραφές στον πίνακα "Orders" που δεν έχουν αντιστοιχίες στους "Customers", αυτές οι παραγγελίες δεν θα εμφανιστούν.
  SELECT Orders.OrderID, Customers.CustomerName
  FROM Orders
  INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

- **LEFT JOIN:** Το LEFT JOIN επιστρέφει όλες τις εγγραφές από τον αριστερό πίνακα (Customers), ακόμη και αν δεν υπάρχουν αντιστοιχίσεις στο δεξί πίνακα (Orders).
  SELECT Customers.CustomerName, Orders.OrderID
  FROM Customers
  LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
  ORDER BY Customers.CustomerName;

- **RIGHT JOIN:** Το RIGHT JOIN επιστρέφει όλες τις εγγραφές από το δεξιό πίνακα (Orders), ακόμα και αν δεν υπάρχουν αντιστοιχίσεις στον αριστερό πίνακα (Customers).
  SELECT Orders.OrderID, Customers.LastName, Customers.FirstName
  FROM Orders
  RIGHT JOIN Orders ON Orders.CustomerID = Customers.CustomerID
  ORDER BY Orders.OrderID;

186. Αναλύστε τη χρήση των συναρτήσεων στον αντικειμενοστραφή προγραμματισμό. Δώστε παράδειγμα πως οι συναρτήσεις καλούνται ως μηνύματα σύνδεσης μεταξύ τάξεων.

Η συνάρτηση (function) είναι ένας συνδυασμός οδηγιών/εντολών συνδυασμένων για να επιτευχθεί κάποιο αποτέλεσμα. Οι συναρτήσεις μπορούν να δεχθούν δεδομένα εισόδου και να δημιουργήσουν αποτέλεσμα επιστροφής. Εάν μια συνάρτηση δεν επιστρέφει ένα αποτέλεσμα, συνήθως ονομάζεται διαδικασία (procedure).

Απ την άλλη, οι μέθοδοι ("member functions") είναι παρόμοιες με τις συναρτήσεις, ανήκουν σε τάξεις ή αντικείμενα και συνήθως εκφράζουν τις λειτουργίες των αντικειμένων / τάξεων. Για παράδειγμα, ένα αντικείμενο τύπου Window συνήθως θα έχει δημόσιες (public) και ιδιωτικές (private) μεθόδους οι οποίες αντιστοιχούν στις λειτουργίες του αντικειμένου που ανήκουν.

Στον αντικειμενοστραφή προγραμματισμό οι μεταβλητές και οι συναρτήσεις που επενεργούν πάνω σε αυτές τις μεταβλητές ομαδοποιούνται σε μια αφαίρεση που ονομάζουμε κλάση.

Παρακάτω δίνεται η κλάση Box και στη συνέχεια φαίνεται πως οι μέθοδοί της αξιοποιούνται στην ανταλλαγή μηνυμάτων – επικοινωνία μεταξύ κλάσεων από την κλάση BoxDemo.

```
class Box {
          double width;
          double height;
          double depth;

          Box(double x, double y, double z) {
               width = x;
               height = y;
               depth = z;
          }

          double volume(){
                         return width * height * depth;
           }
}
```

```
class BoxDemo {
                Box mybox1 = new Box(10, 20, 15);
                Box mybox2 = new Box(3, 6, 9);
                double vol;
                vol = mybox1.volume();
                System.out.println("Volume is " + vol);
                vol = mybox2.volume();
                System.out.println("Volume is " + vol);
}
```

187. Ποια από τα ακόλουθα είναι σωστά και ποια είναι λάθος; Αιτιολογήστε την απάντησή σας για κάθε περίπτωση.

Person p1 = new Student();
Person p2 = new PhDStudent();
PhDStudent phd1 = new Student();
Prof t1 = new Person();
Student s1 = new PhDStudent();

Προκύπτει λογικά ότι η συσχέτιση μεταξύ των ανωτέρω κλάσεων είναι η εξής:

Student *ISA* Person
Prof *ISA* Person
Phd *ISA* Student (επειδή ισχύει Student *ISA* Person, ως εκ τούτου ισχύει PhD *ISA* Person)

Έτσι, λόγω της κληρονομικότητας οι προτάσεις PhDStudent phd1 = new Student(); και Prof t1 = new Person(); είναι σωστές. Απεναντίας, οι προτάσεις Student s1 = new PhDStudent();, Person p1 = new Student(); και Person p2 = new PhDStudent(); είναι λανθασμένες καθώς αντικείμενα κατώτερης τάξης δε μπορούν να καλέσουν constructors ανώτερων τάξεων.

188. Γιατί δεν μπορούμε να αφαιρέσουμε πεδία από μια subclass ενώ μπορούμε να προσθέσουμε;

Από τον ορισμό της υποκλάσης (μια υποκλάση *κληρονομεί και επεκτείνει* τα πεδία και τις μεθόδους της μητέρας κλάσης) δεν είναι δυνατόν η υποκλάση να μην αποτελέσει υπερσύνολο, αναφορικά με τις μεθόδους και τα πεδία της μητέρας κλάσης.
Αρκεί να αναλογιστούμε για παράδειγμα την περίπτωση του *τετραγώνου* και του *κύβου*, όπου εν προκειμένω το *τετράγωνο* είναι η μητέρα κλάση (με δυο τουλάχιστον πεδία μήκος και ύψος) και από την άλλη πλευρά η υποκλάση *κύβος* που επεκτείνει ουσιαστικά το την μητέρα κλάση *τετράγωνο* να μην έχει τουλάχιστον τις δυο επίπεδες διαστάσεις - πεδία μήκος και ύψος αντίστοιχα. Τόσο κατά την αρχική υπόθεση όπου είχε αφαιρεθεί το ένα πεδίο της μητέρας κλάσης (μη κληρονόμιση) *τετράγωνο* όπου για παράδειγμα η κλάση *κύβος* είχε μόνο 2 πεδία π.χ. μήκος –

βάθος, όσο και στην περίπτωση που δε προστίθενται περεταίρω πεδία (μη επέκταση) η κλάση *κύβος* θα ενέπιπτε στη κλάση *τετράγωνο* και ως εκ τούτου δε θα θεωρείτο υποκλάση.

189. Ποια είναι η διαφορά μεταξύ ενός Frame και ενός Dialog;

Ενώ αμφότερα το JFrame και το JDialog κληρονομούν τις ιδιότητες της κλάσης Window, ώστε να μοιράζονται πολλές λειτουργίες, το JFrame είναι ένα κανονικό παράθυρο με (προαιρετικά) κουμπιά και τις διακοσμήσεις, ενώ το JDialog από την άλλη πλευρά δεν έχει κουμπιά μεγιστοποίησης και ελαχιστοποίησης και συνήθως δημιουργείται με στατικές μεθόδους JOptionPane και είναι καλύτερα προσαρμοσμένο στις τροποποιήσεις του αναφορικά με το ότι μπλοκάρει άλλα components μέχρι ο χρήστης να το κλείσει.

190. Ποια είναι η διαφορά μεταξύ ενός αντικείμενου Menu και ενός MenuBar (σε Java);

Το Menu και το MenuBar είναι συστατικά στοιχεία πλοήγησης που υποστηρίζουν διάφορες λειτουργίες και εμφανίζουν επιλογές. Μπορούν να έχουν πολλαπλά επίπεδα υπο-Menu και τα στοιχεία Menu μπορούν είτε να ενεργοποιήσουν τις ενέργειες διακομιστή είτε να χρησιμεύσουν ως σύνδεσμοι σε εξωτερικές διευθύνσεις URL. Η κύρια διαφορά μεταξύ του Menu και του MenuBar είναι ότι το Menu εμφανίζεται κάθετα, ενώ το MenuBar εμφανίζεται οριζόντια. Ακόμη από κατασκευαστικής άποψης το Menu είναι υποκλάση του MenuBar. Έτσι, το MenuBar μπορεί να φιλοξενίσει πολλά Menu και κατ επέκταση MenuItems.

191. Ποια είναι η διαφορά μεταξύ ενός Menu και ενός MenuItem (σε Java);

Το Menu περιέχει τα στοιχεία του μενού. Το MenuItem εμφανίζει την πραγματική επιλογή που μπορεί να επιλέξει ο χρήστης μέσα σε ένα Menu. Τα στοιχεία του Menu (MenuItem-s) προστίθενται στο μενού με τη μέθοδο addMenuItem ().

192. Ποια είναι η διαφορά μεταξύ της επανεκκίνησης και της επαναφόρτωσης ενός Java applet;

Οι λειτουργίες επανεκκίνησης (Restart) και ανανέωσης (Reload) χρησιμοποιούνται για την επανεκκίνηση της εκτέλεσης της εφαρμογής. Η διαφορά μεταξύ αυτών των δύο επιλογών είναι ότι η επαναφόρτωση (Reload) εκφορτώνει την μικροεφαρμογή πριν την επανεκκίνησή της, ενώ η επανεκκίνηση (Restart) όχι. Η επιλογή επαναφόρτωση (Reload) ισοδυναμεί με το κλείσιμο του προγράμματος προβολής μικροεφαρμογών και το άνοιγμα ξανά στην ίδια ιστοσελίδα.

193. Τι είναι οι constructors στην JAVA; Δώστε ένα παράδειγμα.

Μια κλάση περιέχει constructors που καλούνται να δημιουργήσουν αντικείμενα από το πρότυπο της κλάσης. Οι δηλώσεις των constructors μοιάζουν με δηλώσεις μεθόδου - εκτός από το ότι χρησιμοποιούν το όνομα της κλάσης και δεν έχουν τύπο επιστροφής. Για παράδειγμα, το ποδήλατο έχει έναν constructor:

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

Για να δημιουργήσουμε ένα νέο αντικείμενο Bicycle που ονομάζεται myBike, ένας constructor καλείται ως εξής:

```
Bicycle myBike = new Bicycle(30, 0, 8);
new Bicycle(30, 0, 8)
```

Ετσι, ο constructor δεσμεύει χώρο στη μνήμη για το αντικείμενο προετοιμάζοντας τα πεδία του.

194. Να αναφέρετε από ποια μέρη αποτελείται η Java Virtual Machine και ποιες μεθόδους τρέχει.

Μια εικονική μηχανή Java (JVM) είναι μια αφηρημένη υπολογιστική μηχανή που επιτρέπει σε έναν υπολογιστή να εκτελεί ένα πρόγραμμα Java. Υπάρχουν τρία συστατικά του JVM: το specification, το implementation, και το instance. Η προδιαγραφή (specification) είναι ένα έγγραφο που περιγράφει επίσημα τι απαιτείται για μια υλοποίηση της JVM. Έχοντας μια ενιαία προδιαγραφή διασφαλίζει ότι όλες οι υλοποιήσεις (implementations) είναι διαλειτουργικές (interoperable). Μια εφαρμογή JVM (JVM implementation) είναι ένα πρόγραμμα υπολογιστή που πληροί τις απαιτήσεις της προδιαγραφής (specification) JVM. Τέλος το στιγμιότυπο (instance) ενός JVM είναι μια υλοποίηση (implementation) που εκτελείται σε μια διαδικασία που εκτελεί ένα πρόγραμμα υπολογιστή που έχει συνταχθεί σε Java bytecode.

195. Ποιοι είναι οι τύποι μεταβλητών στην Java;

Οι τύποι μεταβλητών (variables) που εντοπίζονται στη Java είναι οι εξής:

a) Local variables
b) Instance variables
c) Class/Static variables

196. Τι κάνει το παρακάτω JAVA πρόγραμμα:int j=5; int k=10; long max=0; max=k>j?k:j;

Η σύνταξη βασίζεται στη δομή ελέγχου: `condition ? first_expression : second_expression;`
Έτσι, ελέγχει αν το k είναι μεγαλύτερο του j και θέτει στο max τη τιμή του k αλλιώς (αν το j είναι μεγαλύτερο) θέτει στο max τη τιμή του j.

197. Ποιος ο ρόλος των εξαιρέσεων στα προγράμματα εισόδου / εξόδου; Δώστε δύο παραδείγματα σε JAVA.

Ο χειρισμός εξαιρέσεων (Exception Handling) είναι η διαδικασία αντίδρασης στην εμφάνιση, κατά τον υπολογισμό, εξαιρέσεων - ανώμαλων ή εξαιρετικών συνθηκών που απαιτούν ειδική επεξεργασία - αλλάζοντας συχνά την κανονική ροή εκτέλεσης του προγράμματος. Συγκεκριμένα, οι εξαιρέσεις εισόδου/εξόδου (IOExceptions) είναι η γενική κατηγορία εξαιρέσεων που παράγονται από αποτυχημένες ή διακοπείσες λειτουργίες εισόδου / εξόδου.

Δυο παραδείγματα χρήσης αυτών παρατίθενται στη συνέχεια όπου εμφανίζονται τα try-catch κατά την εγγραφή σε αρχεία και την ανάγνωση αυτών.

```
import java.io.*;
import static java.nio.file.StandardOpenOption.*;
import java.nio.file.*;
import java.io.*;
import java.nio.charset.Charset;
public class Test {

  public static void main(String[] args) {
      String s = "Hello World! ";
      byte data[] = s.getBytes();
      Path p = Paths.get("C:\\Users\\User\\Desktop\\test.txt");
    write_file(p,data);
    read_file(p);
  }


// Paradeigma xeirismou eggrafhs se arxeio me xrhsh IOException
  public static void write_file(Path p,byte data[]){
      try (OutputStream out = new BufferedOutputStream(
            Files.newOutputStream(p, CREATE, APPEND))) {
               out.write(data, 0, data.length);
      }
      catch (IOException x) {
            System.err.println(x);
        }
  }


// Paradeigma xeirismou anagnwshs apo arxeio me xrhsh IOException
  public static void read_file(Path p){
        Charset charset = Charset.forName("US-ASCII");
```

```
            try    (BufferedReader    reader    =    Files.newBufferedReader(p,
      charset)) {
            String line = null;
              while ((line = reader.readLine()) != null) {
                  System.out.println(line);
            }
            }
            catch (IOException x) {
              System.err.format("IOException: %s%n", x);
            }
        }

      }
```

198. Δημιουργήστε σε JAVA το παιχνίδι κρεμάλα (hangman). Ο χρήστης εισάγει γράμματα τα οποία και συγκρίνονται με λέξεις που επιλέγονται τυχαία από έναν πίνακα με λέξεις string. Μηνύματα επιτυχίας ή αποτυχίας εμφανίζονται στο χρήστη και τον καθοδηγούν για το αποτέλεσμα. Για διευκόλυνσή σας τοποθετήστε στον πίνακα 5 διαφορετικές λέξεις και το πρόγραμμά σας μην επεκταθείτε στη δημιουργία GUI.

```
package hangman;
import java.util.Arrays;
import java.util.Scanner;

public class Hangman{
      public static void main(String[] args) {
      String[] words = {"hello", "world","hungman","wrote","that","program"};
      int randomWordNumber = (int) (Math.random() * words.length);
      char[] enteredLetters = new char[words[randomWordNumber].length()];
      int triesCount = 0;
      boolean wordIsGuessed = false;
      do {
            switch (enterLetter(words[randomWordNumber], enteredLetters)) {
            case 0:
                  triesCount++;
                  break;
                case 1:
                  triesCount++;
                  break;
            case 2:
                  break;
            case 3:
                  wordIsGuessed = true;
                  break;
            }
      } while (! wordIsGuessed);
      System.out.println("\nThe word is " + words[randomWordNumber] +
            " You missed " + (triesCount -findEmptyPosition(enteredLetters)) +
            " time(s)");
      }


      public static int enterLetter(String word, char[] enteredLetters)  {
      System.out.print("(Guess) Enter a letter in word ");
      if (! printWord(word, enteredLetters)){return 3;}
```

```
        System.out.print(" > ");
        Scanner input = new Scanner(System.in);
        int emptyPosition = findEmptyPosition(enteredLetters);
        char userInput = input.nextLine().charAt(0);
        if (inEnteredLetters(userInput, enteredLetters)) {
               System.out.println(userInput + " is already in the word");
               return 2;
        }
        else if (word.contains(String.valueOf(userInput))) {
               enteredLetters[emptyPosition] = userInput;
               return 1;
        }
        else {
               System.out.println(userInput + " is not in the word");
               return 0;
               }
        }


        public static boolean printWord(String word, char[] enteredLetters) {
        boolean asteriskPrinted = false;
        for (int i = 0; i < word.length(); i++) {
               char letter = word.charAt(i);
               if (inEnteredLetters(letter, enteredLetters))
               System.out.print(letter);
               else {
               System.out.print('*');
               asteriskPrinted = true;
               }
        }
        return asteriskPrinted;
        }


        public static boolean inEnteredLetters(char letter, char[] enteredLetters) {
        return new String(enteredLetters).contains(String.valueOf(letter));
        }


        public static int findEmptyPosition(char[] enteredLetters) {
        int i = 0;
        while (enteredLetters[i] != '\u0000') i++;
        return i;
        }
}
```

199. Να ορίσετε σε JAVA την κλάση Tires (Λάστιχα) με τις ακόλουθές ιδιότητες^^ Πλάτος της επιφάνειας του ελαστικού (inches)radius Ακτίνα του ελαστικού (inches)rim Ζαντα (inches)type Τύπος ελαστικού0 - Παντός Καιρού 1 - Ξηρό2 - Βροχή3 - Χιόνι brand Κατασκευαστής model Μοντέλο year έτος κατασκευής Να δημιουργήσετε constructor για την κλάση αυτή, ο ποιος να παίρνει παραμέτρους για όλα τα στοιχεία της κλάσης.

```
public class Tyre {

  private static int width;
  private static int radius;
  private static int rim;
```

```
        private static String type;
        private static String brand;
        private static String model;
        private static int year;

        public Tyre(int wd, int rd, int rm, String tp, String br, String md, int yr) {
               this.width = wd;
               this.radius = rd;
               this.rim = rm;
               this.type = tp;
               this.brand = br;
               this.model = md;
               this.year = yr;
               System.out.println("Tyre created with the following specs:");
               System.out.println("Tyre width= "+ this.width);
                  System.out.println("Tyre radius= "+ this.radius);
               System.out.println("Tyre rim= "+ this.rim);
               System.out.println("Tyre type= "+ this.type);
               System.out.println("Tyre brand= "+ this.brand );
              System.out.println("Tyre model= "+ this.model );
               System.out.println("Tyre year= "+ this.year );

        }

        public static void main(String[] args){

        int the_width=185;
        int the_radius=60;
         int the_rim=18;
        String the_type="snow";
        String the_brand="Michelin";
        String the_model="all_terain_pro";
        int the_year=2017;
         Tyre my_car_tyre= newTyre(the_width,the_radius,the_rim,the_type,the_brand,the_model,the_year);

    }
}
```

200. Ποια είναι η διαφορά μεταξύ μιας abstract class και ενός interface (σε Java);

1) Η abstract κλάση μπορεί να έχει αφηρημένες και μη αφηρημένες μεθόδους. Η interface μπορεί να έχει μόνο αφηρημένες μεθόδους.

2) Η abstract κλάση δεν υποστηρίζει πολλαπλή κληρονομικότητα. Η interface υποστηρίζει πολλαπλή κληρονομικότητα.

3) Η abstract κλάση μπορεί να έχει final, non-final, static και non-static μεταβλητές. Η interface έχει μόνο static και final μεταβλητές.

4) Η abstract κλάση μπορεί να προσφέρει την υλοποίηση της interface. Η interface δεν μπορεί να προσφέρει την υλοποίηση abstract κλάσης.

5) Το keyword “abstract” χρησιμοποιείται για να δηλώσει abstract κλάση. Στο interface, το αντίστοιχο keyword χρησιμοποιείται για να δηλώσει interface.

6) Μια abstract κλάση μπορεί να επεκτείνει μόνο μία κλάση ή μία abstract κλάση τη φορά. Μια interface μπορεί να επεκτείνει οποιοδήποτε αριθμό interface τη φορά

7) Μια abstract κλάση μπορεί να επεκτείνει μια άλλη συγκεκριμένη (κανονική) κλάση ή abstract κλάση. Μια interface μπορεί να επεκτείνει μόνο μια άλλη interface

8) Σε μια abstract κλάση το keyword "abstract" είναι υποχρεωτικό για να δηλωθεί μια μέθοδος ως abstract. Σε ένα interface το keyword "abstract" είναι προαιρετικό για να δηλωθεί μια μέθοδος ως abstract.

9) Μια abstract κλάση μπορεί να έχει protected και public abstract μεθόδους Μια interface μπορεί να έχει μόνο public abstract μεθόδους

201. Τι είναι τα interfaces στην JAVA; Πώς δημιουργούμε ένα interface; Δώστε ένα παράδειγμα.

Η διεπαφή (interface) μοιάζει με κλάση, αλλά δεν είναι κλάση. Μια διεπαφή μπορεί να έχει μεθόδους και μεταβλητές όπως ακριβώς και η κλάση, αλλά οι μέθοδοι που δηλώνονται στη διεπαφή είναι από προεπιλογή αφηρημένες (abstract) (μόνο υπογραφή μεθόδου, χωρίς σώμα). Επίσης, οι μεταβλητές που δηλώνονται σε μια διεπαφή είναι public static και final από προεπιλογή.
Δεδομένου ότι οι μέθοδοι στα interfaces δεν έχουν σώμα, πρέπει να υλοποιηθούν από τη κλάση πριν αποκτηθεί πρόσβαση. Η κλάση που υλοποιεί interface πρέπει να υλοποιήσει όλες τις μεθόδους αυτού του interface.

```
interface MyInterface{
   public void method1();
   public void method2();
}

class Demo implements MyInterface{
   public void method1(){
      System.out.println("implementation of method1");
   }
   public void method2(){
      System.out.println("implementation of method2");
   }
   public static void main(String arg[]){
      MyInterface obj = new Demo();
      obj.method1();
   }
}
```

202. Γράψτε σε Java ένα πρόγραμμα που θα μετατρέπει την θερμοκρασία από βαθμούς Fahrenheit σε Celcius σημείωση: C = 5 (F - 32) / 9"

```
import java.util.Scanner;
public class Fahrenheit {
  public static void main(String[] args) {
      float temperatue;
      Scanner in = new Scanner(System.in);
      System.out.println("Enter temperatue in Fahrenheit");
      temperatue = in.nextInt();
      temperatue = ((temperatue - 32)*5)/9;
      System.out.println("Temperatue in Celsius = " + temperatue);
  }
}
```

203. Γράψτε πρόγραμμα σε Java, το οποίο θα ζητάει από το χρήστη να εισάγει μία λίστα με 6 ονόματα στην αρχική του σειρά, θα ταξινομεί τα ονόματα με αύξουσα αλφαβητική σειρά και κατόπιν θα εμφανίζει ξανά τη λίστα.

```
import java.util.ArrayList;
import java.util.Scanner;

public class Test {

  public static void main(String[] args) {
      ArrayList<String> Names = new ArrayList<String>();
      Scanner in = new Scanner(System.in);
      String name;
      String temp;
      for(int i=0;i<5;i++){
      System.out.print("give name:");
      name= in.next();
      Names.add(name);
      }
      System.out.println("List of names:"+Names);
      for(int i=0;i<Names.size();i++){
      for(int j=i+1;j<Names.size();j++){
            if(Names.get(i).compareTo(Names.get(j))>0){
                  temp=Names.get(i);
                  Names.set(i, Names.get(j));
                  Names.set(j,temp );
            }
      }
      }
      System.out.println("List of names in alphabetical order:"+Names);
  }
}
```

204. Τι είναι το Method Overloading; Δώστε ένα παράδειγμα με χρήση κώδικα Java.

Η μέθοδος "Overloading" είναι μια δυνατότητα που επιτρέπει σε μια κλάση να έχει περισσότερες από μία μεθόδους με το ίδιο όνομα, εάν τα argument lists τους είναι διαφορετικά. Είναι παρόμοιο με την υπερφόρτωση του constructor που επιτρέπει σε μια κλάση να έχει περισσότερους από έναν constructors που έχουν διαφορετικά argument lists.

```
class DisplayOverloading {
      public void disp(char c) {
      System.out.println(c);
      }
      public void disp(char c, int num)  {
      System.out.println(c + " "+num);
      }
}

class Sample{
   public static void main(String args[]) {
      DisplayOverloading obj = new DisplayOverloading();
      obj.disp('a');
      obj.disp('a',10);
   }
}
```

205. Τι είναι το Method Overriding; Δώστε ένα παράδειγμα με χρήση κώδικα Java.

Η δήλωση μιας μεθόδου στην υποκλάση που υπάρχει ήδη στην γονική κλάση είναι γνωστή ως μέθοδος overriding. Η παράκαμψη γίνεται έτσι ώστε μια κλάση παιδιού να μπορεί να δώσει τη δική της implementation σε μια μέθοδο που παρέχεται ήδη από τη μητρική κλάση. Σε αυτήν την περίπτωση, η μέθοδος στην γονική κλάση ονομάζεται overridden μέθοδος και η μέθοδος στην κατηγορία παιδιού ονομάζεται overriding μέθοδος. Στο παρακάτω παράδειγμα η eat() στη κλάση Boy κάνει override την eat() στη κλάση Human.

```
class Human{
   public void eat(){
      System.out.println("Human is eating");
   }
}
class Boy extends Human{
   public void eat(){
      System.out.println("Boy is eating");
   }
   public static void main( String args[]) {
      Boy obj = new Boy();
      obj.eat();
   }
}
```

206. Εξηγήστε αναλυτικά τον παρακάτω κώδικα:

```
interface MyInterface{ public void method1();public void method2(); }
class XYZ implements MyInterface {
  public void method1(){System.out.println(""implementation of method1"");}
  public void method2(){System.out.println(""implementation of method2"");}
  public static void main(String arg[]){MyInterface obj=new XYZ(); obj.method1();}}
```

Αυτός είναι ο τρόπος με τον οποίο μια κλάση υλοποιεί μια διεπαφή. Πρέπει να παρέχει το σώμα όλων των μεθόδων που δηλώνονται στη διεπαφή ή με άλλα λόγια ότι η κλάση πρέπει να εφαρμόσει όλες τις μεθόδους της διεπαφής.

Το interface MyInterface περιέχει τις μεθόδους void method1() και void method2(), των οποίων την υλοποίηση παρέχει η κλάση XYZ (στην ουσία κάνουν από μια εκτύπωση η κάθε μια).

Τέλος η main δημιουργεί ένα αντικείμενο της κλάσης XYZ (το obj) και πάω σε αυτό καλείται η μέθοδος method1() η οποία και τυπώνει το μήνυμα implementation of method1.

207. Δημιουργήστε (σε Java) το class inventory χρησιμοποιώντας Vector Arrays μέσα στο inventory όπου εμπεριέχονται αντικείμενα τύπου string[ ] με χαρακτηριστικά όπλων π.χ mithril sword, Plate Armour κτλ.

```
package inventory;

import java.util.Enumeration;
import java.util.Scanner;
import java.util.Vector;

public class Inventory {
      private  String inventory_item;

      public Inventory(String item){
           this.inventory_item=item;
             System.out.println(   this.inventory_item+"   added   to   the
inventory");
      }


      public static void main(String[] args) {
      Scanner SC = new Scanner(System.in);
         System.out.print("Define invenotry capacity: ");
      int invenotry_size = SC.nextInt();
      String item;
         Vector<Inventory>          invenotry_list          =          new
Vector<Inventory>(invenotry_size);
      for(int i=0; i<invenotry_size;i++){
             System.out.print("\nset item to be adde to the inventory: ");
             item = SC.next();
          Inventory new_item=new Inventory(item);
```

```
                invenotry_list.addElement(new_item);
        }


           System.out.print("\n\nInventory contains: ");
        for(int i=0; i<invenotry_size;i++){
                 System.out.print(invenotry_list.get(i).inventory_item+ ", ");
        }

        }

}
```

208. Δημιουργήστε (σε Java) το πρόγραμμα Print(), που εκτυπώνει μέσω κληρονομικότητας στην οθόνη το αποτέλεσμα των τάξεων:
sum, που προσθέτει αριθμούς integer,
sub, που αφαιρεί αριθμούς integer και
multiply που πολλαπλασιάζει αριθμούς integer

```
package calculator;


import java.lang.Math;
import java.util.Scanner;

class Summary{
      public static int add(int a, int b){
      return Math.addExact(a, b);
      }
}

class Subtract extends Summary{
      public static int sub(int a, int b){
      int c=b*(-1);
      return add(a, c);
      }
}

class Multiplication extends Summary{
      public static int multi(int a, int b){
      int result=0;
      for(int i=0;i<b;i++){result=add(a, result);}
      return result;
      }
}


public class Calculator {

      private static int x;
      private static int y;
```

```
	public Calculator(int a, int b){
		this.x=a;
		this.y=b;
	}

	public static void main(String[] args) {
		int factor_1,factor_2;
		Scanner SC = new Scanner(System.in);
		System.out.print("\nset factor 1: ");
		factor_1= SC.nextInt();
		System.out.print("\nset factor 2: ");
		factor_2= SC.nextInt();
		Calculator result= new Calculator(factor_1,factor_2);
		print(result);
	  }

	public static void print(Calculator input){
	    System.out.println("sum="+Summary.add(input.x, input.y));
	    System.out.println("sub="+Subtract.sub(input.x, input.y));
	    System.out.println("mlt="+Multiplication.multi(input.x, input.y));
	}



}
```

209. Δημιουργήστε (σε Java) την τάξη Rectangle, την τάξη Square και την τάξη Circle η κάθε μία περιέχει τουλάχιστον 3 μεταβλητές και μια συνάρτηση που υπολογίζει το εμβαδόν τους. Οι τάξεις Square και Circle κληρονομούν από την τάξη Rectangle τις μεταβλητές της, υπολογίζουν και εκτυπώνουν το εμβαδόν τους (εμβαδό κύκλου = 2*3.14*R*R, εμβαδό τετραγώνου = Width*Width).

```
package shapes;

public class Rectangle{
public int width = 0;
public int height = 0;
	public static String color[];
public Rectangle(int w, int h, String cl[]){
		width = w;
		height = h;
		System.arraycopy(color, 0, cl, 0, cl.length);
}
public int getArea(){
		return width * height;
}
}
-----------------------------------------------------------------------------------------
public class Square extends Rectangle{
```