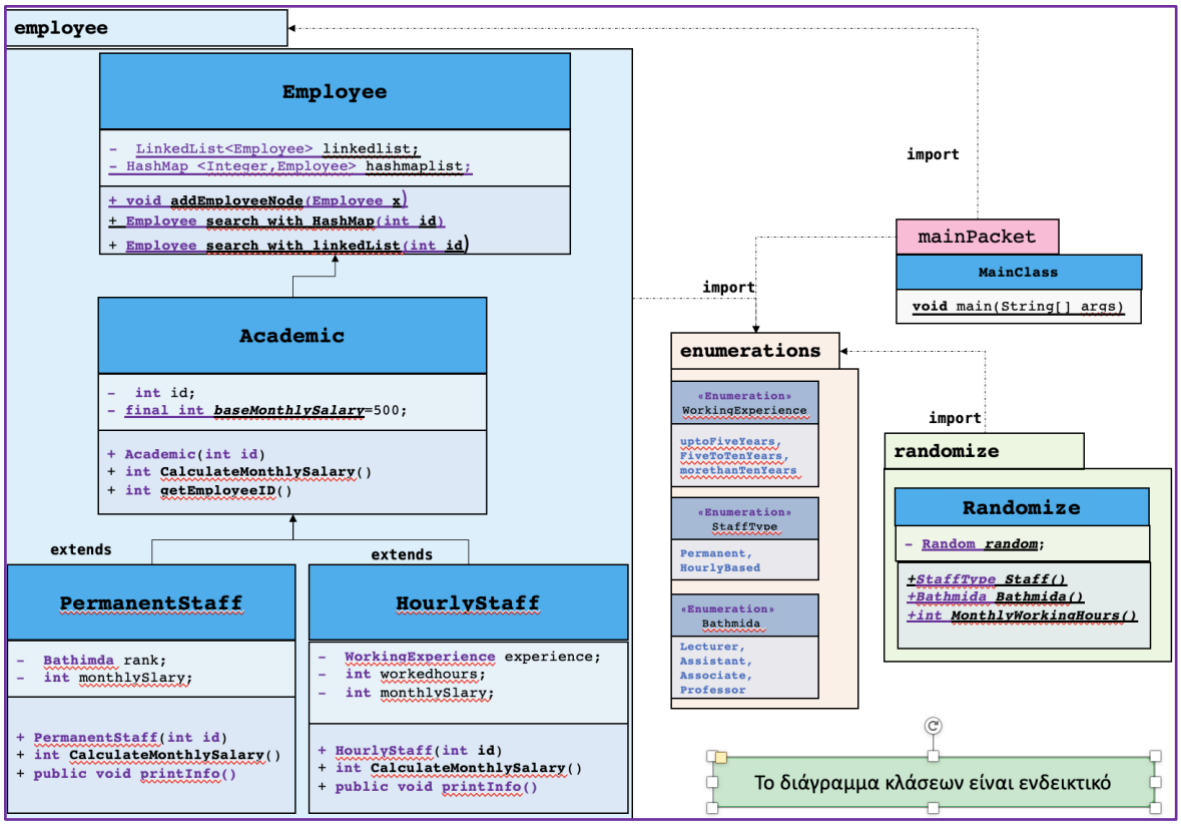
Άσκηση 5η

Ο σκοπός της 5ης άσκησης είναι η απόκτηση εμπειρικής γνώσης σχετικά με τη χρήση της πολλαπλής κληρονομικότητας (inheritance), των αφηρημένων κλάσεων, του γενερικού προγραμματισμού, την αναβάθμιση και υποβάθμιση αντικειμένων (down-casting, up-casting) και τη χρήση συλλογών (HashMap, LinkedList).

Περιγραφή του προβλήματος: Επεκτείνουμε την 4η άσκηση προσθέτοντας την κλάση Employees η οποία είναι στην κορυφή του δέντρου κληρονομικότητας και υλοποιεί: *α)* μία συλλογή διασυνδεόμενης λίστας (LinkedList) η οποία περιέχει το σύνολο των εργαζομένων που δημιουργούνται κατά την εκτέλεση του προγράμματος, και *β)* μια συλλογή ζευγών κλειδιού-τιμής (key-value pairs) HashMap που περιέχει το σύνολο των εργαζομένων που δημιουργούνται κατά την εκτέλεση του προγράμματος το οποίο περιλαμβάνει και τον μοναδικό τους αριθμό (ID) (*συμβουλευτείτε το διάγραμμα κλάσεων και θα δείτε ότι το ID είναι πεδίο της κλάσης Academic*).



Ζητούμενα: Εκτελέστε το πρόγραμμα που προσομοιώνει την μισθοδοσία 100.000 υπαλλήλων. Στη συνέχεια αναζητήστε έναν συγκεκριμένο υπάλληλο και εκτυπώστε τα στοιχεία του. Δημιουργήστε στην κλάση Employees τις στατικές συναρτήσεις:

α) ***void addEmployeeNode (Employee x)***

β) ***Employee search_with_HashMap (int id)***

γ) ***Employee search_with_linkedList(int id)***

Μπορείτε τις παραπάνω συναρτήσεις να τις καλείτε μέσω της *main* όπως φαίνεται στο παράδειγμα πιο κάτω. Για το σκοπό αυτό χρησιμοποιείστε όπου θεωρείτε απαραίτητο την αναβάθμιση και υποβάθμιση αντικειμένων καθώς και τη δήλωση αφηρημένων μεθόδων.

Ενδεικτικό Κώδικας της Κλάσης MainClass:

```
public class MainClass {
    public static void main(String[] args) {
        System.out.println("HELLO PROJECT WEEK 5");
        int nOf=100000;
        createRandomizedEmployees(nOf);

        int searchemployeeID=50000;
        System.out.println("\n\nSearch with Linked List for Employee with id:"+searchemployeeID);
        Employee x=Employee.search_with_linkedList(searchemployeeID);
        x.printInfo();

        System.out.println("\nSearch with Hash Map with id:"+searchemployeeID);
        x=Employee.search_with_HashMap(searchemployeeID);
        x.printInfo();
    }
    public static void createRandomizedEmployees(int k) {
        System.out.println("Creating "+ k +" Randomized Employees");
        for(int i=1;i<=k;i++) {
            StaffType staff=Randomize.Staff();
            if (staff==StaffType.Permanent) {
                PermanentStaff pstaff=new PermanentStaff(i);
                Employee.addEmployeeNode((Employee)pstaff);//UP-CASTING
            }

            if (staff==StaffType.HourlyBased) {
                HourlyBasedStaff hstaff=new HourlyBasedStaff(i);
                Employee.addEmployeeNode((Employee)hstaff);//UP-CASTING
            }
        }
        System.out.println("Finished Randomized Employees");
    }
}
```

Μετρήστε και συγκρίνεται τον χρόνο που χρειάζεστε να βρείτε τον υπάλληλο που αναζητάμε μέσω του HashMap και μέσω της LinkedLIst. Μπορείτε για αυτό το σκοπό να κάνετε χρήση της ***System.currentTimeMillis().***

Ενδεικτικό Κώδικας της Κλάσης Employee:

```
package employees;
import java.util.HashMap;
import java.util.LinkedList;

abstract public class Employee {
    static private LinkedList<Employee> linkedlist=new LinkedList<Employee>();
    static private HashMap <Integer,Employee> hashmaplist = new HashMap<Integer,Employee>();

    abstract public void printInfo();
    abstract public int getEmployeeID();

    public static void addEmployeeNode(Employee x) {
        linkedlist.add(x);
        hashmaplist.put( x.getEmployeeID(),x);
    }

    public static Employee search_with_HashMap(int id) {
        long startTime,estimatedTime;
        Employee tmp=null;
        startTime = System.currentTimeMillis();
```