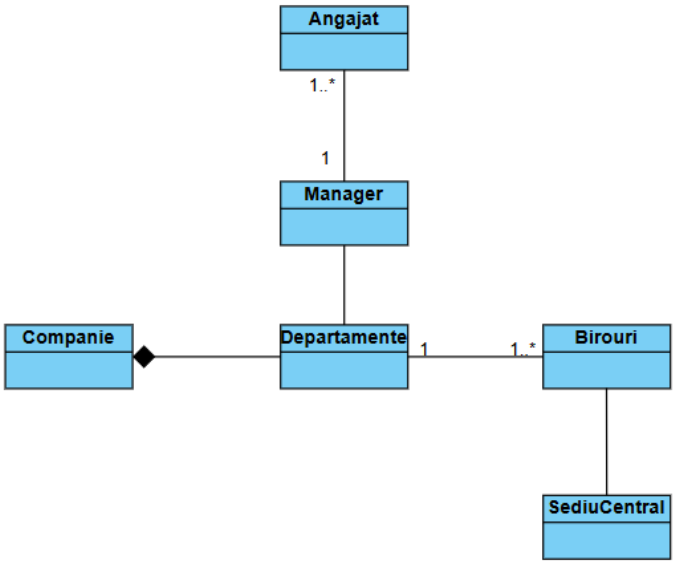
Laborator 6

4.



Main.java | Companie.java | Departament.java | SediuCentral.java | Birou.java | Mana

```java
package lab6;

import java.util.Arrays;

class Main {
    public static void main(String[] args) {
        Angajat a1 = new Angajat("George Enescu");
        Angajat a2 = new Angajat("Ana Pop");
        Angajat a3 = new Angajat("Mihai Eminescu");

        Manager manager1 = new Manager("Ion Popescu", Arrays.asList(a1, a2));
        Manager manager2 = new Manager("Maria Ionescu", Arrays.asList(a3));

        Birou b1 = new Birou("Cluj");
        Birou b2 = new Birou("Bucuresti");
        SediuCentral sediu = new SediuCentral("Bucuresti - Central");

        Departament d1 = new Departament("IT", manager1, Arrays.asList(b1, sediu));
        Departament d2 = new Departament("HR", manager2, Arrays.asList(b2));

        Companie companie = new Companie("TechCorp", Arrays.asList(d1, d2));
        companie.afiseazaStructura();
    }
}
```

Main.java | Companie.java | Departament.java | SediuCentral.java | Birou.j

```java
package lab6;

import java.util.List;

class Companie {
    private String nume;
    private List<Departament> departamente;

    public Companie(String nume, List<Departament> departamente) {
        this.nume = nume;
        this.departamente = departamente;
    }

    public void afiseazaStructura() {
        System.out.println("Compania: " + nume);
        for (Departament d : departamente) {
            System.out.println("  Departament: " + d.getNume());
            System.out.println("    Manager: " + d.getManager().getNume());
            System.out.println("    Angajati in echipa: ");
            for (Angajat a : d.getManager().getAngajati()) {
                System.out.println("      - " + a.getNume());
            }
            for (Birou b : d.getBirouri()) {
                System.out.println("    Birou: " + b.getLocatie());
            }
        }
    }
}
```

Main.java | Companie.java | Departament.java | SediuCentral.java | Birou.java

```java
package lab6;

import java.util.List;

class Departament {
    private String nume;
    private Manager manager;
    private List<Birou> birouri;

    public Departament(String nume, Manager manager, List<Birou> birouri) {
        this.nume = nume;
        this.manager = manager;
        this.birouri = birouri;
    }

    public String getNume() {
        return nume;
    }

    public Manager getManager() {
        return manager;
    }

    public List<Birou> getBirouri() {
        return birouri;
    }
}
```

Main.java | Companie.java | Departament.java | Se

```java
package lab6;

class SediuCentral extends Birou {
    public SediuCentral(String locatie) {
        super(locatie);
    }
}
```

Main.java Companie.java Departan

```java
package lab6;

class Birou {
    private String locatie;

    public Birou(String locatie) {
        this.locatie = locatie;
    }

    public String getLocatie() {
        return locatie;
    }
}
```

Main.java Companie.java Departament.java SediuCentra

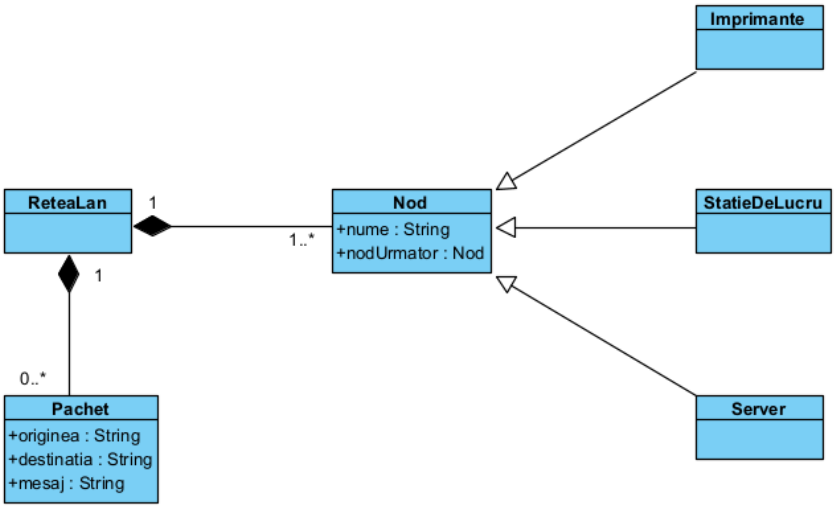
```java
package lab6;

import java.util.List;

class Manager extends Angajat {
    private List<Angajat> angajati;

    public Manager(String nume, List<Angajat> angajati) {
        super(nume);
        this.angajati = angajati;
    }

    public List<Angajat> getAngajati() {
        return angajati;
    }
}
```

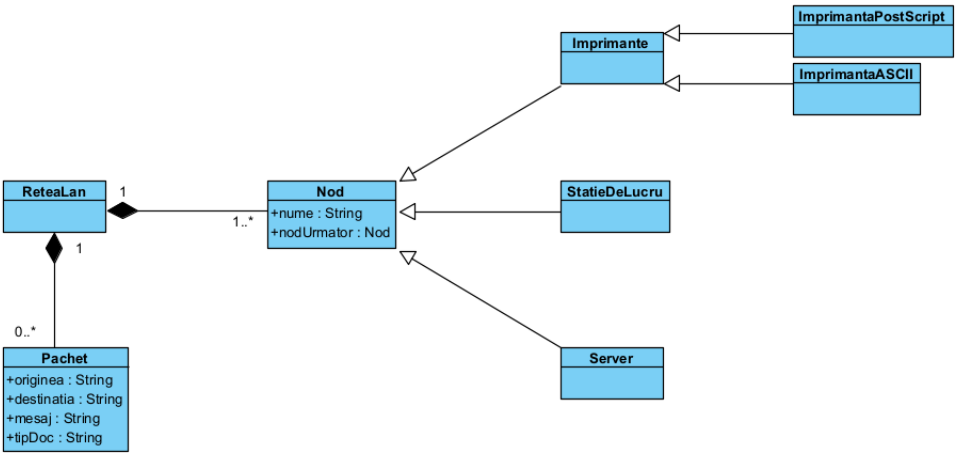
Main.java Companie.java Depar

```java
package lab6;

class Angajat {
    protected String nume;

    public Angajat(String nume) {
        this.nume = nume;
    }

    public String getNume() {
        return nume;
    }
}
```

```
Compania: TechCorp
  Departament: IT
    Manager: Ion Popescu
    Angajati in echipa:
      - George Enescu
      - Ana Pop
    Birou: Cluj
    Birou: Bucuresti - Central
  Departament: HR
    Manager: Maria Ionescu
    Angajati in echipa:
      - Mihai Eminescu
    Birou: Bucuresti
```

5.






```java
package lab6b;

public class ReteaLan {
    private Nod nodStart;

    public ReteaLan(Nod nodStart) {
        this.nodStart = nodStart;
    }

    public void transmitePachet(Pachet pachet) {
        Nod curent = nodStart;
        do {
            curent.proceseazaPachet(pachet);
            curent = curent.getNodUrmarator();
        } while (curent != null && curent != nodStart);
    }
}
```

```java
package lab6b;

public abstract class Nod {
    protected String nume;
    protected Nod nodUrmarator;

    public Nod(String nume) {
        this.nume = nume;
    }

    public void setNodUrmarator(Nod nod) {
        this.nodUrmarator = nod;
    }

    public Nod getNodUrmarator() {
        return nodUrmarator;
    }

    public String getNume() {
        return nume;
    }

    public abstract void proceseazaPachet(Pachet pachet);
}
```

```java
package lab6b;

public class Pachet {
    private String originea;
    private String destinatia;
    private String mesaj;
    private String tipDoc;

    public Pachet(String originea, String destinatia, String mesaj, String tipDoc) {
        this.originea = originea;
        this.destinatia = destinatia;
        this.mesaj = mesaj;
        this.tipDoc = tipDoc;
    }

    public String getOriginea() {
        return originea;
    }

    public String getDestinatia() {
        return destinatia;
    }

    public String getMesaj() {
        return mesaj;
    }

    public String getTipDoc() {
        return tipDoc;
    }
}
```

```java
package lab6b;

public class StatieDeLucru extends Nod {
    public StatieDeLucru(String nume) {
        super(nume);
    }

    @Override
    public void proceseazaPachet(Pachet pachet) {
        System.out.println("Statia " + nume + " trimite pachetul: " + pachet.getMesaj());
    }
}
```

```java
package lab6b;

public class Server extends Nod {
    public Server(String nume) {
        super(nume);
    }

    @Override
    public void proceseazaPachet(Pachet pachet) {
        if (nume.equals(pachet.getDestinatia())) {
            System.out.println("Serverul " + nume + " a primit mesajul: " + pachet.getMesaj());
        }
    }
}
```

```java
package lab6b;

public abstract class Imprimanta extends Nod {
    public Imprimanta(String nume) {
        super(nume);
    }

    public abstract boolean acceptaTip(String tipDoc);

    @Override
    public void proceseazaPachet(Pachet pachet) {
        if (nume.equals(pachet.getDestinatia()) && acceptaTip(pachet.getTipDoc())) {
            System.out.println("Imprimanta " + nume + " a tiparit mesajul: " + pachet.getMesaj());
        }
    }
}
```

```java
package lab6b;

public class ImprimantaASCII extends Imprimanta {
    public ImprimantaASCII(String nume) {
        super(nume);
    }

    @Override
    public boolean acceptaTip(String tipDoc) {
        return tipDoc.equalsIgnoreCase("ASCII");
    }
}
```

Main.java | Pachet.java | Nod.java | StatieDeLucru.java | Server.java | Imprimanta.java

```java
package lab6b;

public class ImprimantaPostScript extends Imprimanta {
    public ImprimantaPostScript(String nume) {
        super(nume);
    }

    @Override
    public boolean acceptaTip(String tipDoc) {
        return tipDoc.equalsIgnoreCase("ASCII") || tipDoc.equalsIgnoreCase("PostScript");
    }
}
```

6.

Ex1.

A – nu este generalizare intre Carte si Produs, este Realizare

B - nu este generalizare intre Carte si Produs, este Realizare

C – Produs nu este o clasa, este o interfata

D -Corect

E –Clasa Carte defineste o compozitie de obiecte de tip Capitol

Ex2.

A – nu este extends, se foloseste implements pentru ca Produs e o interfata

B – clasa si interfata sunt scrise gresit, corect este class Revista implements Produs { ... }

C - Corect

D – in interfata nu scri vizibilitatea functiilor, raname doar: float getPret();

E - in interfata nu scri vizibilitatea functiilor, raname doar: float getPret();

F - Corect

G – functia nu trebuie sa returneze nimic

Ex3.

A - Corect

B – trebuia scris private int grade;

C - Corect

D – nu este public, este private

E – nu are tipul int de returnare

F –clasa Proiect nu extinde clasa, aceasta contine un obiect de tip Student

G - Corect

H – trebuia sa fie o lista de diagrame

Ex4.

A – clasa StudentBursier morsteneste Student

B - Corect

C –nu este public, este private

D - Corect

E –nu este private, este public

F - Corect

Ex5.

A –clasa profesor nu extinde clasa Materie

B - Corect

C – clasa Materie nu are Profesor ca atribut

D –clasa Materie nu are un vector de materi

Ex6.

A - Corect

B – este unidirectionala

C –clasa Materie contine o lista de Teste

D - Corect

E - Corect

Ex7.

A – Materie contine o lista de Laboratoare, nu extinde clasa

B - Materie contine o lista de Laboratoare, nu extinde clasa

C - Corect

D – clasa Materie contine o lista de laboratoare

E – nu creaza o colectie noua in clasa Materie

Ex8.

A – intre profesor si materie este asociere unidirectionala

B – asocierea este de la Profesor la Materie

C – clasele Test, Curs si Laborator nu extind clasa Materie, clasa Materie nu e superclasa

D - Corect

E - clasele Test, Curs si Laborator nu extind clasa Materie, clasa Materie contine o lista de obiecte din acele clase

Ex9.

A – corect este class Carte implements Produs

B - Corect

C –corect este private float pret;

D – corect este float getPret();

E - Corect

F – pret nu este functie

G - Corect

H – functia nu returneaza Produs

I - Corect

Ex10.

A – clasa nu este abstracta

B - Corect

C –functiile abstracte nu se implementeaza

D –functia nu este abstracta

Ex11.

A - Corect

B –Angajat este o clasa nu interfata

C – nu este corect, trebuia public abstract void calculPlata();

D - Corect

Ex12.

A –clasa Rezervare nu extinde Client

B –nu specifica static

C - Corect

D - Corect

E - Corect

F –functia nu este statica

Ex13.

A – clasa Proiect defineste un agregat de tip Student

B - clasa Proiect defineste un agregat de tip Student

C - Corect

D - clasa Proiect defineste un agregat de tip Student

E - Corect