SUB A

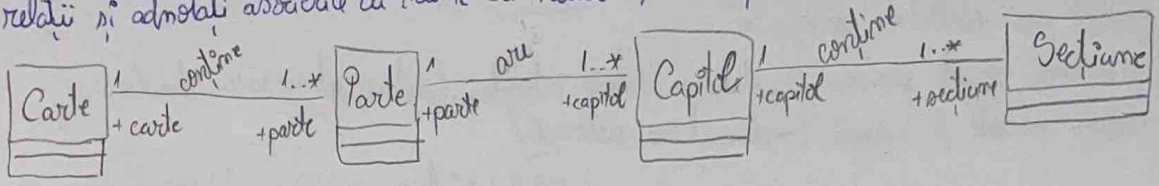
10

1. Care este utilitatea realizării diagramelor de interacțiune în etapa de analiză a unui sistem soft?

Identificarea atributelor și rafinarea descrierii comportamentului acestora.

usecase —> diagramă de interacțiune
diagram

2. Realizați o diagramă de clase care să repr. o carte: „O carte e compusă dintr-un nr. de părți, fiecare parte conținând un nr. de capitole; capitolele constau din secțiuni". Focusați-vă doar pe clase și relații și adnotați asocierile cu nume de roluri și multiplicități.



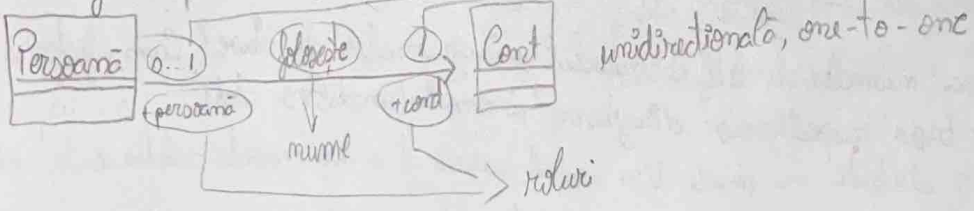
eventual să pun și atribute

3. Care sunt regulile de reprezentare a numelor de roluri și multiplicităților capetelor de asociere în codul sursă? Dați exemple (diagramă și cod)

Atributele care fac referință la clasa asociată trb. să aibă același nume ca numele de roluri de pe diagramă.

Multiplicități:

one: referință simplă la clasa asociată
many: colecție de referințe la clasa asociată




```
public class Persoană
{ private Cont cont;
  public Persoană() { cont = new Cont(); }
  public Cont getCont() { return cont; }
}
```

1. Care sunt diferențele dintre modelul structural de analiză și cel corespunzător etapei de proiectare a unui sistem?

Sunt diferențe în cea ce privește nivelul de detalii și obiectivele.

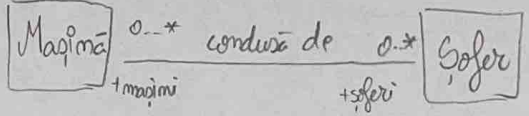
modelul structural de analiză:
- concentrat pe înțelegerea și descrierea sistemului în termeni de elem. structurali și interacțiuni dintre acestea
- identificare entități cheie, relații și funcții ale sistemului
- comportament general al sistemului
- imagine clară a ceea ce trb. realizat

etapa de proiectare:
- detalierea arhitecturii și a componentelor identificate la analiză
- nivel mai detaliat de specificații
- abordează aspecte precum structura modulară, interfețe, algoritmi, optimizări, tehnologii
- soluții concrete pt. a pregăti implementarea sistemului

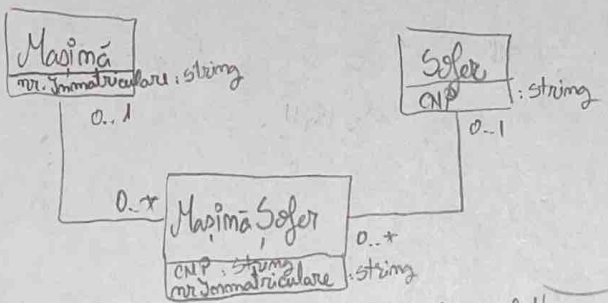
2. O asociere de tip many-to-many la nivelul modelului trebuie descompusă? Care este alternativa la descompunerea uzuală și în ce condiții se justifică? Exemplu.

- trebuie descompusă folosind o clasă de legătură ce referă clasele implicate în relație
- alternativă: asocieri calificate

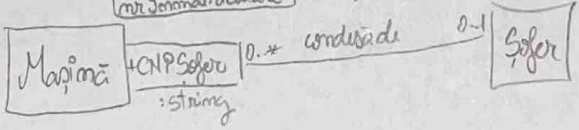
relația de descompus:



1. uzual:

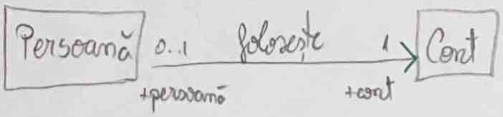


2. asocieri calificate:



3. Care este utilitatea numelor de rol la nivelul diagramelor de clase? Cum afectează acestea codul sursă scris pe baza respectivelor diagrame? Exemplu (model vs. cod)

numele de rol devin atribute în clasele din codul sursă pt. a implementa relația dintre clasele în cauză

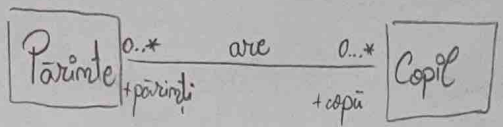



```
public class Persoană {
    private Cont cont;
    public Persoană() { cont = new Cont(); }
    public getCont() { return cont; }
}
```

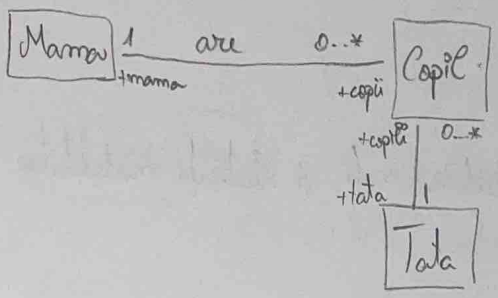
1. Ce credeți că diferențiază ingineria programării de ramurile inginerești clasice (înafară de natura produsului final)?

1. abordare abstractă și virtuală (lucru cu concepte matematice, algoritmi, SO) vs concentrarea pe obiecte fizice, materiale
2. flexibilitate și rapiditate în modificări vs modificări probabil lente și costisitoare
⋮

2. Realizați o diagramă de clase care să reprezinte relațiile dintre părinți și copii (adnotați asocierile cu nume de roluri și multiplicități)



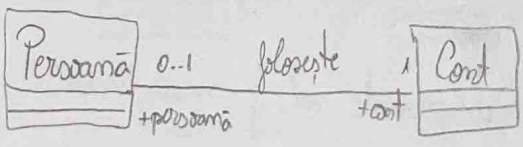
Sau



3. Care sunt proprietățile unei relații de asociere între clase și cum se reflectă în codul sursă scris pe baza unei diagrame de clase?

- nume
- roluri
- multiplicități

Numele de roluri trebuie să fie ca în diagramă.
Multiplicități: 1: avem o referință simplă spre cealaltă clasă (asociată)
0..*: avem un set de referințe spre cealaltă clasă (asociată)



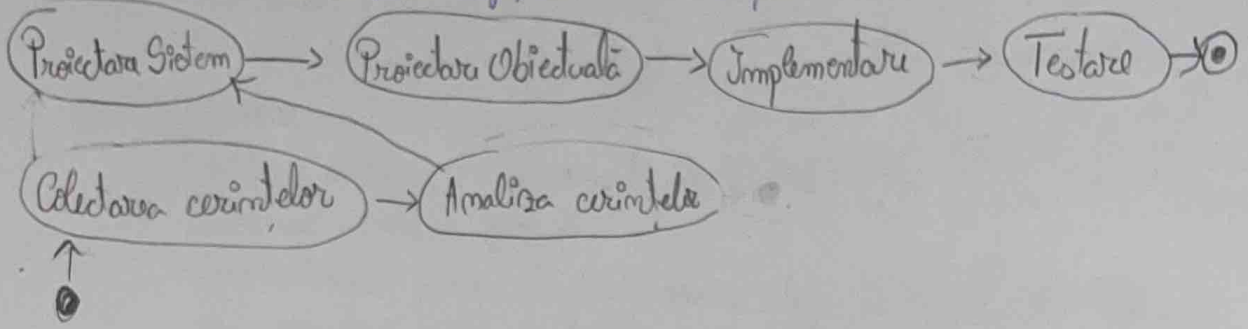

```
public class Persoană {
    private Cont cont;
    public Persoană() { this.cont = new Cont; }
    public getCont() { return this.cont; }
```

}

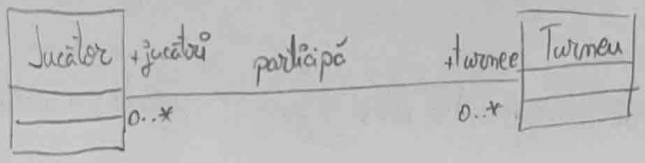
1. Argumentați necesitatea realizării modelelor în procesul de dezvoltare a softului

Realizarea modelelor în procesul de dezvoltare software este necesară pentru a clarifica cerințele clientului și a le structura într-o aplicație într-un mod optim. De asemenea, în cazul în care un proiect se întinde pe o perioadă lungă de timp, în care programatorii se schimbă, modelele vor face înțelegerea și mentinerea aplicației mult mai ușoară.

2. Realizați o diagramă de activități care să ilustreze etapele tehnice ale dezvoltării softului, în eventualitatea în care acestea par desfășura strict secvențial.



3. Dați un exemplu de asociere many-to-many bidirecțională și ilustrați modalitatea de transformare a acesteia în cod sursă.




```
public class Jucator {
    public ICollection<Turneu> turnee { get; private set; } = new HashSet
                                                <Turneu>();}

public class Turneu {
    public ICollection<Jucator> jucători { get; private set; } = new HashSet
                                                <Jucator>(); }
```

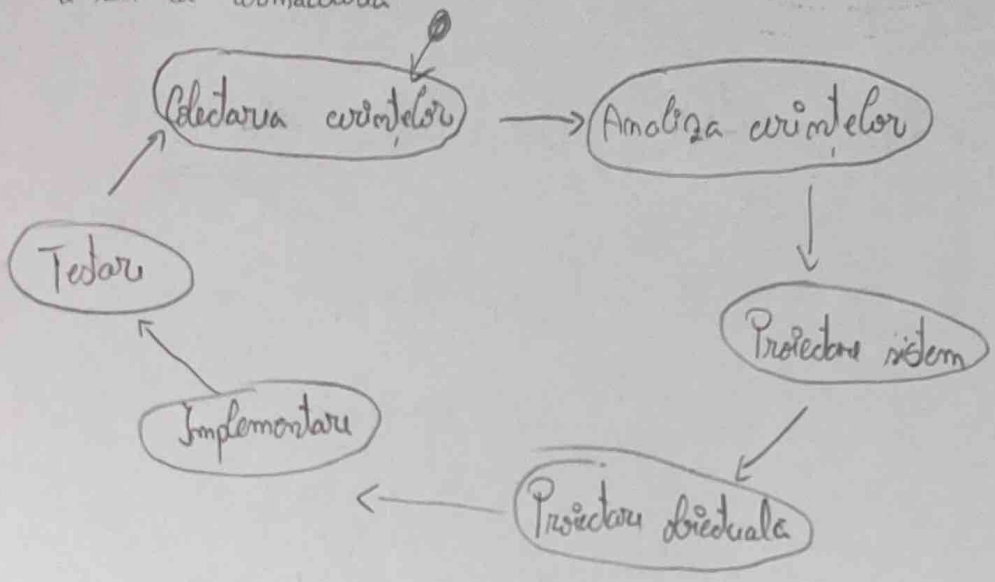
1. Ce diferențiază etapa de analiză de etapa de colectare a cerințelor unui sistem soft?

etapa de colectare a cerințelor: diagrama cazurilor de utilizare (cea cu culori)

etapa de analiză: diagrame de clase și de interacțiune

2. Realizați o diagramă de activități care să ilustreze etapele tehnice ale dezv. softului, în eventualitatea în care acestea s-ar desfășura incremental.

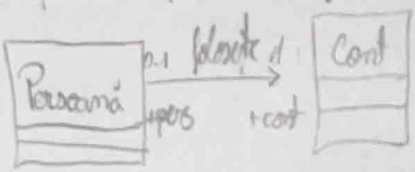
- o parte a unui sistem este analizată, proiectată, implementată și testată complet înainte de a trece la următoarea



3. Ce înțelegeți prin asociere unidirecțională, respectiv bidirecțională între 2 clase? Dați exemple și explicați diferențele la nivelul codului sursă aferent.

bidirecțional

unidirecțional: doar primul are ref. la al doilea




```
public class Persoana {
    public Cont cont { get; private set } = new Cont();
}

public class Cont {
}
```

15

1. Explicați diferența dintre etapa de analiză și cea de proiectare din ciclul de dezv. a softului.

analiză: diograme de clase, interacție
proiectare: ~~diagrame de interacțiune~~

2. Realiz. o diagramă de interacțiune care să corespundă comandării telefonice a unei pizza, din mom ridicării telefonului până la consumarea acesteia. Introduceți și activitățile efectuate de terți.