Tècnica del backtracking

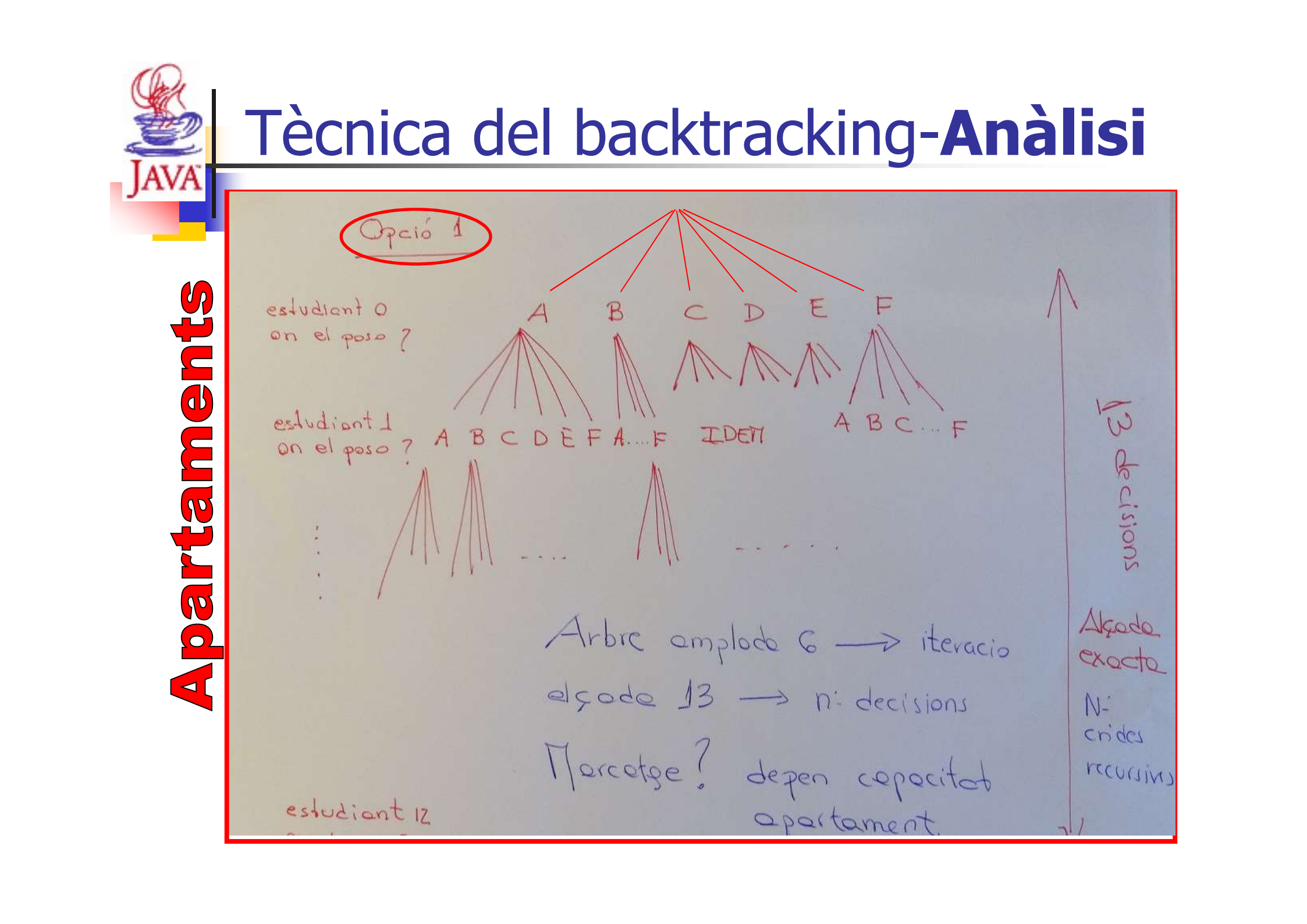
Exemple 2

Apartaments

- Usant **backtracking**:

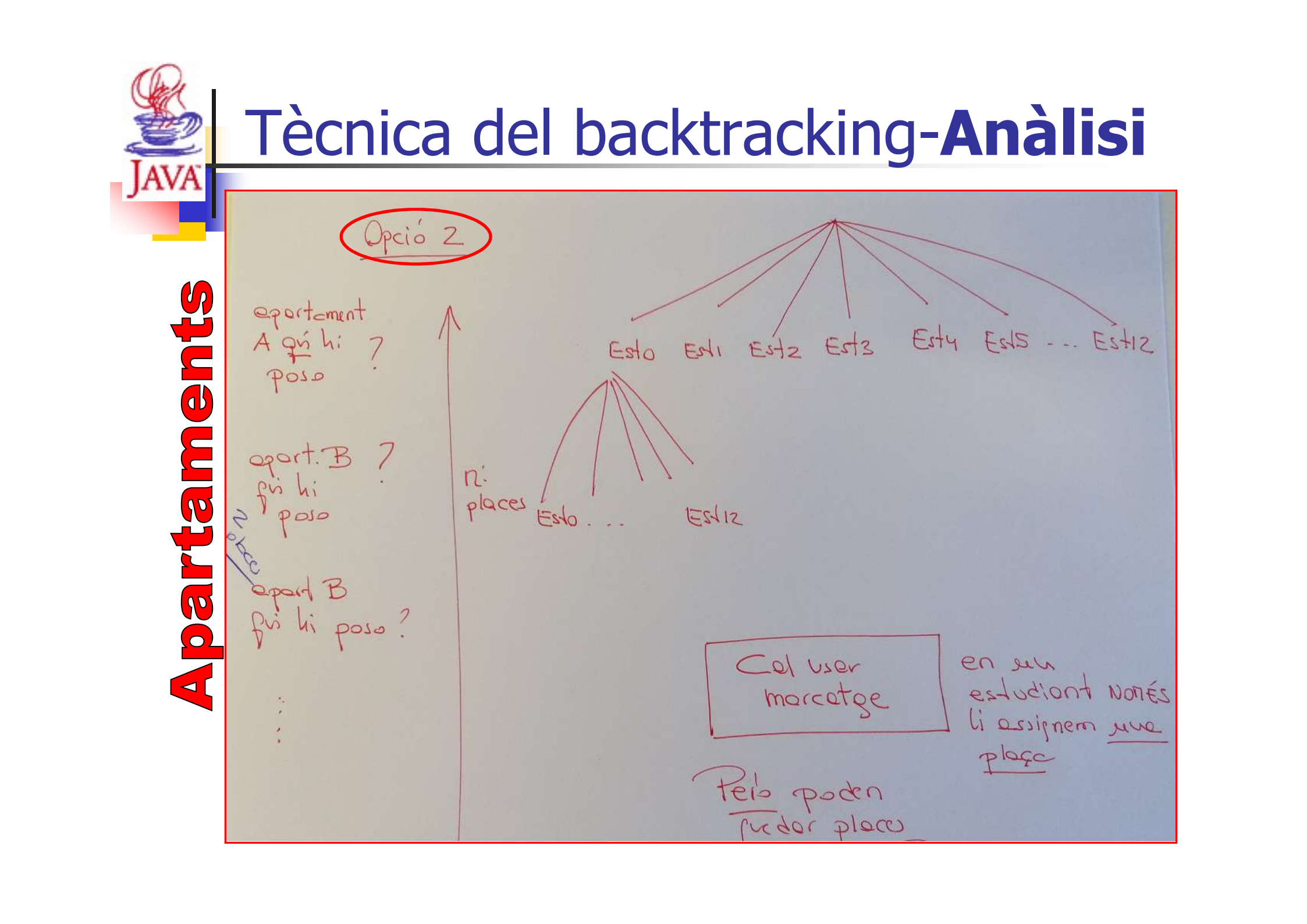
- Els monitors del viatge de fi de curs d'un grup de batxillerat han de distribuir els estudiants en els apartaments que han llogat, ens demanen de confeccionar un programa per determinar **l'assignació d'un apartament a cadascun d'ells**.
- Posen a la seva disposició 6 mini-apartaments, anomenats A, B, C, D, E i F i són 13 els estudiants a distribuir. La cabuda en cadascun dels apartaments són respectivament la següent: 1, 2, 3, 2, 3, 4.

- Fes un programa per a cadascun dels següents plantejaments:
  - 1.- Troba totes les distribucions possibles d'estudiants en apartaments.
  - 2.- Troba una única distribució.



Tècnica del backtracking-**Anàlisi**

Apartaments

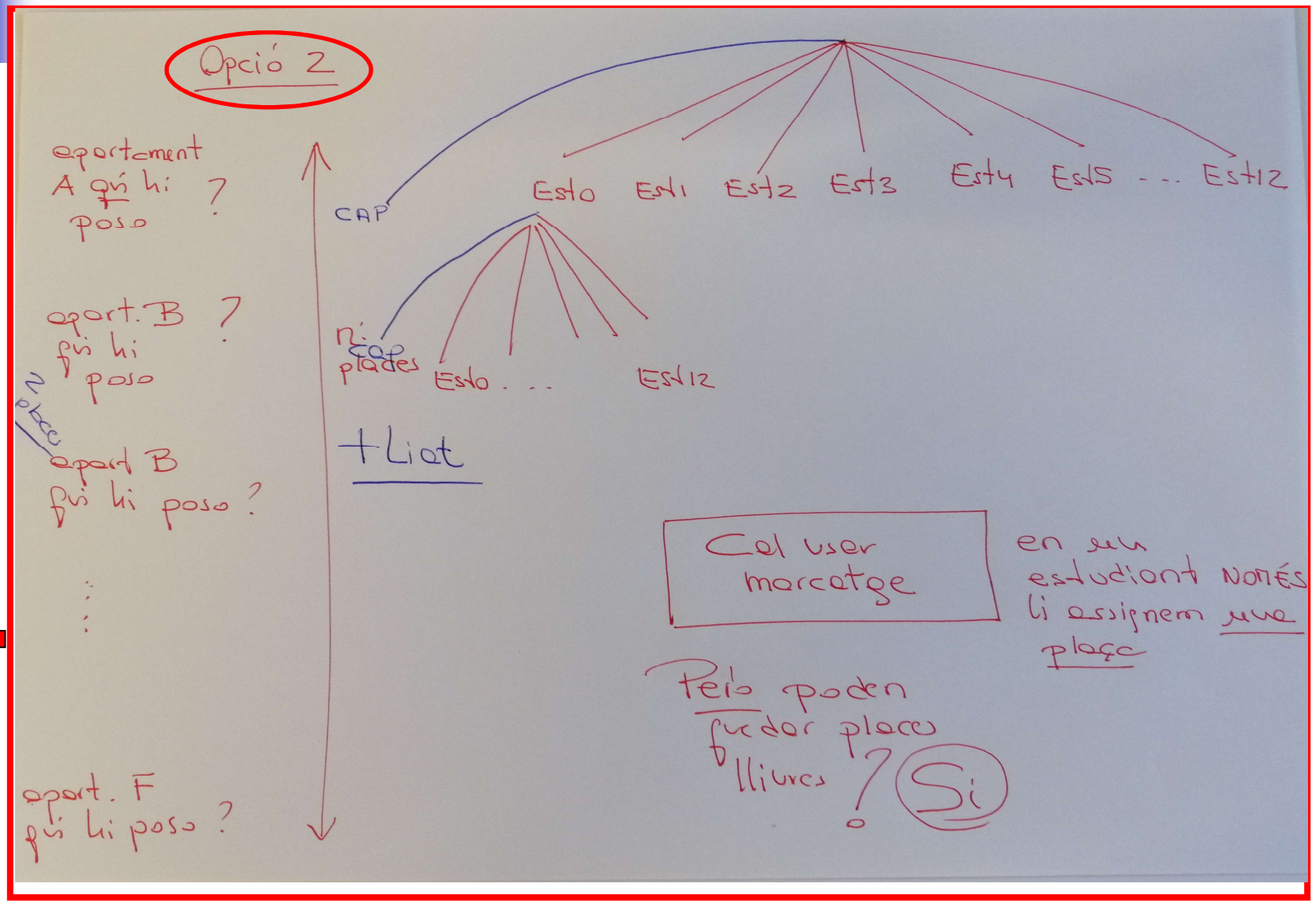


Tècnica del backtracking-**Anàlisi**

Apartaments

Tècnica del backtracking-**Anàlisi**

Apartaments



Tècnica del backtracking-**Anàlisi**

Apartaments

- **Decisió**: assignar a cada estudiant un apartament. Nivell 0 en quin apartament ubiquem l'estudiant 0?.....
- **Quantes decisions**: Exacte - 13 decisions
- **Acceptable**: si l'estudiant cap a l'apartament seleccionat. Portar el control per a cadascun dels apartaments de les places lliures que té.
- **Solució**: quant tots els estudiants tinguin apartament assignat.
- **Completable**: queden estudiants sense apartament. Amb les dades del problema sabem que sempre hi ha solució.
- **Espai de cerca**: arbre d'alçada 13 (nombre d'estudiants) i amplada 6 (nombre d'apartaments).
- **Esquema a aplicar**: totes les solucions.

Crides recursives

while

Tècnica del backtracking-Solució

Apartament.java

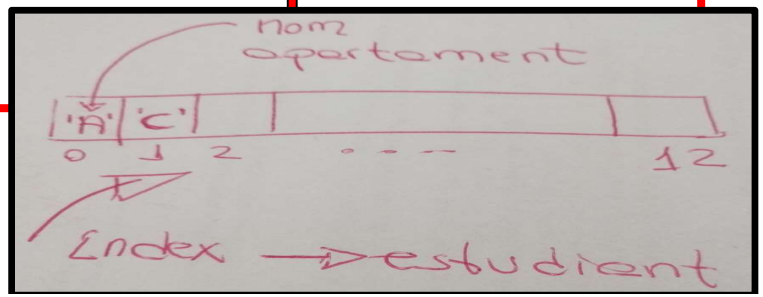
```
public class Apartament{
   private char nom; //identificador
   private int capacitatActual; //si està buit és
   // la màxima de l'apartament
   public Apartament(char nom, int capacitat){
       this.nom=nom; capacitatActual=capacitat;
   }
   public char getNom(){return nom;}
   public int getCapacitatActual(){
       return  capacitatActual;}
   public void addCapacitatActual(){
       capacitatActual++;}
   public void remCapacitatActual(){
       capacitatActual--;}
} // fi classe
```

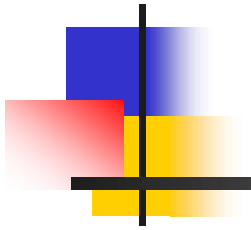
Tècnica del backtracking-Solució

Solució-java

```
public class Solucio{
   private static Apartament apartaments[]=
                                   /*Sentències*/
   public static void main(String args[]){
                /*Sentències*/
    }
  private static void BackTotesSolucions(char [] sol,
  int k){
       /*Sentències*/
   }
} // fi classe
```

les decisions les ubiquem en una taula de caràcters que serà l'identificador de l'apartament.





Programació Avançada

Solució

Totes les solucions

Tècnica del backtracking-Solució

Solució a Java

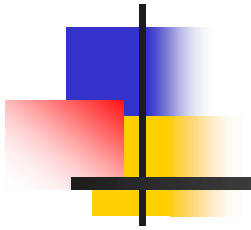
```
public class Solucio{
   private static  Apartament apartaments[]=
   {   new Apartament('A',1), new Apartament('B',2),
       new Apartament('C',3), new Apartament('D',2),
       new Apartament('E',3), new Apartament('F',4)
   }
   public static void main(String args[]){
       char []sol=new char [13]; //solució
       BackTotesSolucions(sol, 0);
   }
```

Cada cop trobem una solució la visualitzem a pantalla

Tècnica del backtracking-Solució

Solució-java

```
private static void BackTotesSolucions(char [] sol,
int k){ ← nivell - estudiant
    int j=0; //seleccionem el primer apartament
    while (j<6){ → amplada-apartament
    //és accep assignar al estu k-èssim l'apartament j-èssi
        if (apartaments[j].getCapacitatActual() !=0){
                apartaments[j].remCapacitatActual();
                sol[k]=apartaments[j].getNom();
                if (k==12) //solucio
                        VisualitzarResultat(sol);
                else BackTotesSolucions(sol,k+1);
                sol[k]='  '; //treiem el valor- No cal !!!
                apartaments[j].addCapacitatActual();
        } // fi if acceptable
        j++; //passem al següent valor
   }//fi while
} // fi procediment
```



Programació Avançada

Variant. Cal fer-ne l'adaptació

Tècnica del backtracking-Variant

Solucio-java

```
public class Solucio{                          fora !
   private          static          Apartament
   apartaments[ ]=/*Sentències*/
   public static void main(String args[]){
                /*Sentències*/
    }
  private static void BackTotesSolucions(char [] sol,
   int k){
                /*Sentències*/
   }
} // fi classe
```

Backtracking-Solució Variant

solucio_java

```
public class Solucio{
   private Apartament apartaments[ ]=
   {new Apartament('A',1), new Apartament('B',2),
   new Apartament('C',3), new Apartament('D',2),
   new Apartament('E',3), new Apartament('F',4)
   };
   public static void main(String args[]){
      char []sol=new char [13]; //solució
      Solucio s= new Solucio();
      s.BackTotesSolucions(sol, 0);
}
```

Cal crear un objecte per poder usar un atribut NO estàtic, és a dir atribut propi de cada objecte

Tècnica: Backtracking-Solució

Solució-java

```
private static void BackTotesSolucions(char [] sol,
   int k){
    int j=0; //seleccionem el primer apartament
    while (j<6){
   //és accepab assignar a l'est k-èssim l'apartament j-èssi
        if (apartaments[j].getCapacitatActual() !=0){
              apartaments[j].remCapacitatActual();
              sol[k]=apartaments[j].getNom();
              if (k==12) //solucio
                    VisualitzarResultat(sol);
              else BackTotesSolucions(sol,k+1);
              sol[k]=' '; //treiem el valor- No cal !!!
              apartaments[j].addCapacitatActual();
        } // fi if acceptable
        j++; //passem al següent valor
   }//fi while
} // fi procediment
```

fora !

Els mètodes estàtics **no** poden accedir als atributs no estàtics de la classe

Programació Avançada

```
public static boolean Back1Solucio( TaulaSolucio  TS, int k){
   boolean trobada=false;
   inicialitzem_valors_domini_decisio_k
   agafar_el_primer_valor
   while (quedin_valors && !trobada){
        if (valor_acceptable){  //no viola les restriccions
               anotem_el_valor_a_la_solucio
               if (solucio_final) trobada=true;
               else if (solucio_completable)
                      trobada=Back1Solucio(TS, k+1);
               if (!trobada) desanotem_el_valor
        } //fi if
        agafar_seguent_valor
         //passem al següent germà a la dreta
   } // fi while
   return trobada;
} // fi procediment
```

Esquema de cerca

Important

Opcionalment es pot fer el tractament abans d'acabar el procediment

Adaptació a 1 solució
Esquema 1 solució

Indicar les implementacions que fan l'adaptació correctament

Tècnica: Backtracking-Solució

Proposta 1

```
public class Variant1{
    private static Apartament apartaments[]={
        new Apartament('A',1), new Apartament('B',2), new
        Apartament('C',3), new Apartament('D',2), new
        Apartament('E',3), new Apartament('F',4)
    }
    public static void main(String args[]){
            char []sol=new char [13]; //solució
            if (!Back1Solucio(sol, 0))
            System.out.println("No hi ha solució");
            else VisualitzarResultat(sol);
    }
    private static void VisualitzarResultat(char []sol){
        // sentències
    }
```

Proposta 1

```
private static boolean Back1Solucio(char [] sol, int k){
      int j=0; //seleccionem el primer apartament
      boolean trobat=false;
      while (j<6 && !trobat){
            //és acceptable assignar al alumne k-èssim al
            // apartament j-èssim ? només si hi cap !!!
            if (apartaments[j].getCapacitatActual() !=0){
                  apartaments[j].remCapacitatActual();
                  sol[k]=apartaments[j].getNom();
                  if (k==12) //solucio
                        trobat=true;
                  else trobat=Back1Solucio(sol,k+1);
                  if (!trobat){
                        sol[k]=''; //treiem el valor
                        apartaments[j].addCapacitatActual();
                  }
            } // fi if
            j++; //passem al següent valor
      }//fi while
      return trobat;
} // fi procediment
```

Proposta 1

```
private static boolean Back1Solucio(char [] sol, int k){
      int j=0; //seleccionem el primer apartament
      boolean trobat=false;
      while (j<6 && !trobat){
            //és acceptable assignar al alumne k-èssim al
            // apartament j-èssim ? només si hi cap !!!
            if (apartaments[j].getCapacitatActual() !=0){
                  apartaments[j].remCapacitatActual();
                  sol[k]=apartaments[j].getNom();
                  if (k==12) //solucio
                        trobat=true;
                  else trobat=Back1Solucio(sol,k+1);
                  if (!trobat){
                        sol[k]='’'; //treiem el valor
                        apartaments[j].addCapacitatActual();
                  }
            } // fi if
            j++; //passem al següent valor
      }//fi while
      return trobat;
} // fi procediment
```

Tècnica: Backtracking-Solució

Proposta 2

```
public class Variant2{
      private static Apartament apartaments[]={
            new  Apartament('A',1),  new  Apartament('B',2),  new
            Apartament('C',3),    new    Apartament('D',2),    new
            Apartament('E',3), new Apartament('F',4)
      }
      public static void main(String args[]){
                  char []sol=new char [13]; //solució
                  if (!Back1Solucio(sol, 0))
                  System.out.println("No hi ha solució");
      }
```

Proposta 2

```
private static boolean Back1Solucio(char [] sol, int k){
            int j=0; //seleccionem el primer apartament
            boolean trobat=false;
            while (j<6 && !trobat){
            //és acceptable assignar al alumne k-èssim al
            //apartament j-èssim ? només si hi cap !!!
              if (apartaments[j].getCapacitatActual() !=0){
                      apartaments[j].remCapacitatActual();
                      sol[k]=apartaments[j].getNom();
                      if (k==12) { //solucio
                           VisualitzarResultat(sol);
                           trobat=true;
                      }
                      else  trobat=Back1Solucio(sol,k+1);
                      if (!trobat){
                           sol[k]='';  //treiem el valor
                           apartaments[j].addCapacitatActual();
                      }
                } // fi if
                j++; //passem al següent valor
            }//fi while
            return trobat;
} // fi procediment
```

Proposta 2

```
private static boolean Back1Solucio(char [] sol, int k){
        int j=0; //seleccionem el primer apartament
        boolean trobat=false;
        while (j<6 && !trobat){
        //és acceptable assignar al alumne k-èssim al
        //apartament j-èssim ? només si hi cap !!!
          if (apartaments[j].getCapacitatActual() !=0){
                apartaments[j].remCapacitatActual();
                sol[k]=apartaments[j].getNom();
                if (k==12) { //solucio
                     VisualitzarResultat(sol);
                     trobat=true;
                }
                else  trobat=Back1Solucio(sol,k+1);
                if (!trobat){
                     sol[k]='';  //treiem el valor
                     apartaments[j].addCapacitatActual();
                }
           } // fi if
           j++; //passem al següent valor
        }//fi while
        return trobat;
} // fi procediment
```

Tècnica: Backtracking-Solució

Proposta 3

```
public class Variant3{
      private static Apartament apartaments[]={
            new  Apartament('A',1),  new  Apartament('B',2),  new
            Apartament('C',3),    new    Apartament('D',2),    new
            Apartament('E',3), new Apartament('F',4)
      }
      public static void main(String args[]){
            char []sol=new char [13]; //solució
            if (!Back1Solucio(sol, 0))
                  System.out.println("No hi ha solució");
            else VisualitzarResultat(sol);
      }
```

Proposta 3

```
private static boolean Back1Solucio(char [] sol, int k){
      int j=0; //seleccionem el primer apartament
      boolean trobat=false;
      while (j<6 && !trobat){
      //és acceptable assignar al alumne k-èssim al
      //apartament j-èssim ? només si hi cap !!!
            if (apartaments[j].getCapacitatActual() !=0){
                  apartaments[j].remCapacitatActual();
                  sol[k]=apartaments[j].getNom();
                  if (k==12)  //solucio
                        trobat=true;
                  else trobat=Back1Solucio(sol,k+1);
                  sol[k]='''; //treiem el valor  !!
                  apartaments[j].addCapacitatActual();
            } // fi if
            j++; //passem al següent valor
      }//fi while
      return trobat;
} // fi procediment
```

Proposta 3

```
private static boolean Back1Solucio(char [] sol, int k){
      int j=0; //seleccionem el primer apartament
      boolean trobat=false;
      while (j<6 && !trobat){
      //és acceptable assignar al alumne k-èssim al
      //apartament j-èssim ? només si hi cap !!!
            if (apartaments[j].getCapacitatActual() !=0){
                  apartaments[j].remCapacitatActual();
                  sol[k]=apartaments[j].getNom();
                  if (k==12)  //solucio
                        trobat=true;
                  else trobat=Back1Solucio(sol,k+1);
                  sol[k]='’; //treiem el valor  !!
                  apartaments[j].addCapacitatActual();
            } // fi if
            j++; //passem al següent valor
      }//fi while
      return trobat;
} // fi procediment
```

Tècnica: Backtracking-Solució

Proposta 4

```
public class Variant4{
      private static Apartament apartaments[]={
            new  Apartament('A',1),  new  Apartament('B',2),  new
            Apartament('C',3),    new    Apartament('D',2),    new
            Apartament('E',3), new Apartament('F',4)
      }
      public static void main(String args[]){
            char []sol=new char [13]; //solució
            if (!Back1Solucio(sol, 0))
            System.out.println("No hi ha solució");
      }
```

Proposta 4

```
private static boolean Back1Solucio(char [] sol, int k){
      int j=0; //seleccionem el primer apartament
      boolean trobat=false;
      while (j<6 && !trobat){
      //és acceptable assignar al alumne k-èssim al
      //apartament j-èssim ?només si hi cap !!!
            if (apartaments[j].getCapacitatActual() !=0){
                  apartaments[j].remCapacitatActual();
                  sol[k]=apartaments[j].getNom();
                  if (k==12) { //solucio
                        VisualitzarResultat(sol);
                              trobat=true;
                  }
                  else trobat=Back1Solucio(sol,k+1);
                  sol[k]='’'; //treiem el valor  !!!
                  apartaments[j].addCapacitatActual();
            } // fi if
            j++; //passem al següent valor
      }//fi while
      return trobat;
} // fi procediment
```

Proposta 4

```
private static boolean Back1Solucio(char [] sol, int k){
      int j=0; //seleccionem el primer apartament
      boolean trobat=false;
      while (j<6 && !trobat){
      //és acceptable assignar al alumne k-èssim al
      //apartament j-èssim ?només si hi cap !!!
            if (apartaments[j].getCapacitatActual() !=0){
                  apartaments[j].remCapacitatActual();
                  sol[k]=apartaments[j].getNom();
                  if (k==12) { //solucio
                        VisualitzarResultat(sol);
                              trobat=true;
                  }
                  else trobat=Back1Solucio(sol,k+1);
                   sol[k]='’'; //treiem el valor  !!!
                  apartaments[j].addCapacitatActual();
            } // fi if
            j++; //passem al següent valor
      }//fi while
      return trobat;
} // fi procediment
```

Tècnica: Backtracking-Solució

Proposta 5

```
public class Variant5{
      private static Apartament apartaments[]={
            new  Apartament('A',1),  new  Apartament('B',2),  new
            Apartament('C',3),    new    Apartament('D',2),    new
            Apartament('E',3), new Apartament('F',4)
      }
      public static void main(String args[]){
            char []sol=new char [13]; //solució
            boolean trobat=false;
            Back1Solucio(sol, 0, trobat);
            if (!trobat) System.out.println("No hi ha solució");
            else   VisualitzarResultat(sol);
      }
```

Proposta 5

```
private static void Back1Solucio(char [] sol, int k,
boolean trobat){
      int j=0; //seleccionem el primer apartament
      while (j<6 && !trobat){
      //és acceptable assignar al alumne k-èssim al
      // apartament j-èssim ? només si hi cap !!!
        if (apartaments[j].getCapacitatActual() !=0){
                apartaments[j].remCapacitatActual();
                sol[k]=apartaments[j].getNom();
                if (k==12)  //solucio
                      trobat=true;
                else Back1Solucio(sol,k+1,trobat);
                if (!trobat){
                      sol[k]='';  //treiem el valor
                      apartaments[j].addCapacitatActual();
                }
        } // fi if
        j++; //passem al següent valor
      }//fi while
} // fi procediment
```

Proposta 5

```
private static void Back1Solucio(char [] sol, int k,
boolean trobat){
      int j=0; //seleccionem el primer apartament
      while (j<6 && !trobat){
      //és acceptable assignar al alumne k-èssim al
      // apartament j-èssim ? només si hi cap !!!
        if (apartaments[j].getCapacitatActual() !=0){
                apartaments[j].remCapacitatActual();
                sol[k]=apartaments[j].getNom();
                if (k==12)  //solucio
                      trobat=true;
                else Back1Solucio(sol,k+1,trobat);
                if (!trobat){
                      sol[k]='';  //treiem el valor
                      apartaments[j].addCapacitatActual();
                }
        } // fi if
        j++; //passem al següent valor
      }//fi while
} // fi procediment
```

Tècnica: Backtracking-Solució

Proposta 6

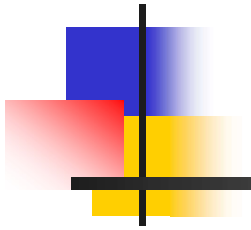
```
public class Variant6{
      private static Apartament apartaments[]={
            new  Apartament('A',1),  new  Apartament('B',2),  new
            Apartament('C',3),    new    Apartament('D',2),    new
            Apartament('E',3), new Apartament('F',4)
      }
      private static boolean trobat;
      public static void main(String args[]){
            char []sol=new char [13]; //solució
            trobat=false;
            Back1Solucio(sol, 0);
            if (!trobat) System.out.println("No hi ha solució");
            else   VisualitzarResultat(sol);
      }
```

Proposta 6

```
private static void Back1Solucio(char [] sol, int k){
      int j=0; //seleccionem el primer apartament
      while (j<6 && !trobat){
      //és acceptable assignar al alumne k-èssim al
      // apartament j-èssim ? només si hi cap !!!
        if (apartaments[j].getCapacitatActual() !=0){
                apartaments[j].remCapacitatActual();
                sol[k]=apartaments[j].getNom();
                if (k==12)  //solucio
                     trobat=true;
                else  Back1Solucio(sol,k+1);
                if (!trobat){
                     sol[k]='';  //treiem el valor
                     apartaments[j].addCapacitatActual();
                }
         } // fi if
        j++; //passem al següent valor
      }//fi while
} // fi procediment
```

Proposta 6

```
private static void Back1Solucio(char [] sol, int k){
      int j=0; //seleccionem el primer apartament
      while (j<6 && !trobat){
      //és acceptable assignar al alumne k-èssim al
      // apartament j-èssim ? només si hi cap !!!
        if (apartaments[j].getCapacitatActual() !=0){
                apartaments[j].remCapacitatActual();
                sol[k]=apartaments[j].getNom();
                if (k==12)  //solucio
                      trobat=true;
                else  Back1Solucio(sol,k+1);
                if (!trobat){
                      sol[k]='';  //treiem el valor
                      apartaments[j].addCapacitatActual();
                }
         } // fi if
        j++; //passem al següent valor
      }//fi while
} // fi procediment
```



Programació Avançada

Adaptar per **trobar la Millor Solució**.

1.- Aquella que deixi més apartaments buits, sense cap estudiant, i a més

2.- Que deixi en diferents apartaments a l'estudiant 2 i l'estudiant 5, per incompatibilitat de caràcters

Restricció

Tècnica: Backtracking-Solució

Apartament-Java

```
public class Apartament{
   private char nom;
   private int capacitatActual;
   private int capacitatMax; //Nou
   public Apartament(char nom, int capacitat){
               capacitatMax=capacitat; //Nou
               this.nom=nom; capacitatActual=capacitat;
   }
   public char getNom(){return nom;}
   public int getCapacitatActual(){
               return  capacitatActual;}
   //Afegit
   public int getCapacitatMax(){return capacitatMax;}
   public void addCapacitatActual(){
        capacitatActual++;}
   public void remCapacitatActual(){
        capacitatActual--;}
} // fi classe
```

Si són iguals vol dir que està buit

Tècnica: Backtracking-Solució

Solucio-java

```
public class Solucio{
   private       static                    Apartament
   apartaments[]=.. //Idem

   public static void main(String args[]){
       char []sol=new char [13]; //solució
       //Nou
       char[]millor= new char [13];
       int qLliuresMi[]={-1};
       // Afegim paràmetres
       BackMillorSolucions(sol, 0, millor,qLliuresMi);
     VisualitzarResultat(millor);
   }
```

Tècnica: Backtracking-Solució

Solució-Java

```
private static void BackMillorSolucions(char [] sol,
int k,char [ ]millor, int qM[ ]){
    int j=0; //seleccionem el primer apartament
    while (j<6){
       if (apartaments[j].getCapacitatActual() !=0){
          apartaments[j].remCapacitatActual();
          sol[k]=apartaments[j].getNom();
          if (k==12 && sol[2]!=sol[5]){ //solucio
                  if (millor(qM)) //mètode
                     for (int i=0; i<=12; i++)
                       millor[i]=sol[i];
          }
          else if (k<12) //tots no apartament!!
                BackMillorSolucions(sol,k+1, millor, qM);
          sol[k]=' '; //treiem el valor- No cal !!!
          apartaments[j].addCapacitatActual();
          ...Idem
```

Perquè?

Tècnica: Backtracking-Solució

Mètode privat

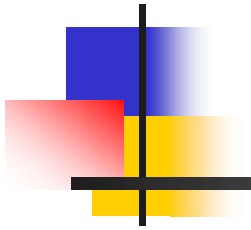
```
private static boolean millor(int []qM){
   int cont=0;
   for (int i=0; i<=5; i++){
       if (apartaments[i].getCapacitatActual()
                ==apartaments[i].getCapacitatMax())
              cont++;
   } //fi for
   if (cont>qM[0]){
       qM[0]=cont;
       return true;
   }
   return false;
}
```

És adient programar mètodes privats per cada part de l'esquema, millorant així la compensió i la descomposició funcional

Millorem eficiència

Portar un comptador d'apartaments lliures

Exercici



Programació Avançada

Generalització
A
N estudiants
M apartaments de X capacitat

Dades especificades per l'usuari de l'aplicació