Structures de données linéaires

Exemples :

- Tableaux
- Listes chaînées
- Files
- Piles

Structures de données

On distingue

- La **spécification**

(par exemple pour pile :

vider(pile) → _

empiler(elt, pile) → _

sommet(pile) → elt

dépiler(pile) → _)

Structures de données

On distingue

- La **spécification**

(comment utiliser la structure,

de l'extérieur)

Structures de données

On distingue

- L' **implémentation**

(par exemple pour une pile :

```
Class pileDentiers {
    private int pile[] =new int[99];
}
```

)

Structures de données

On distingue

- L' **implémentation**

(par exemple pour une pile :

```
Class pileDentiers {
    private int pile[] =new int[99];
}
```

)

Structures de données

On distingue

- L' **implémentation**

(comment c'est fait dedans)

Structures de données

On distingue

- L' **implémentation** écrite à partir des structures de données existantes

Structures de données

On distingue

- L' **implémentation** écrite à partir des structures de données existantes :

  - multiples attributs

  - tableaux statiques

  - chaînage

Structures de données attributs multiples

```
public class complexe {

 public double partieRéelle()
 public double partieImaginaire()

}
```

Structures de données attributs multiples

```
public class complexe {

 private double module;
 private double argument;

 public double partieRéelle() {
    return(module*Math.cos(argument));
 }

 public double partieImaginaire() {
    return(module*Math.sin(argument));
 }
}
```

Structures de données tableaux statiques

```
public class complexe {

 private double[] coord=new double[1];


 public double partieRéelle() {
    return(coord[0]);
 }

 public double partieImaginaire() {
    return(coord[1]);
 }
}
```

Structures de données tableaux statiques

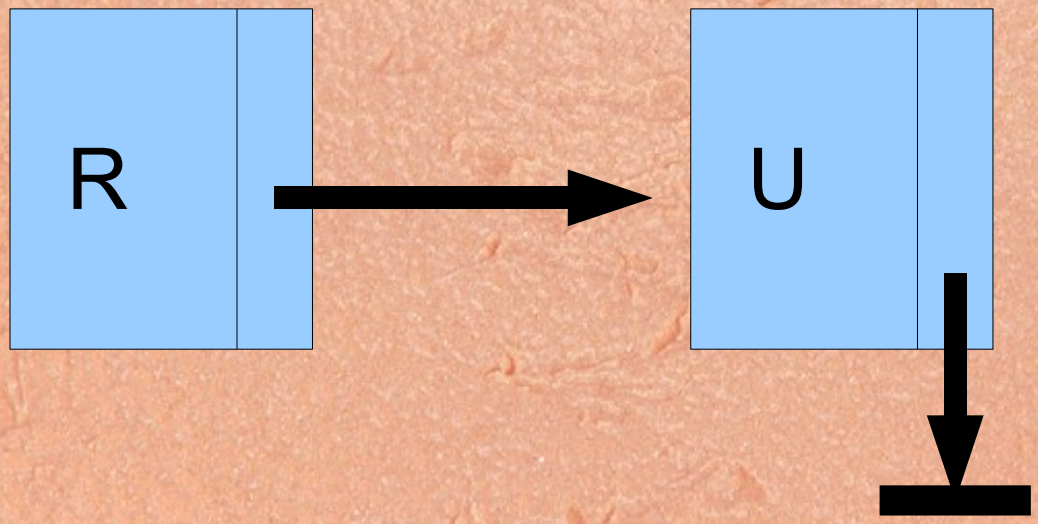
```
public class chaineCaracteres {

 private char[] texte=new char[255];

}
```

Structures de données chaînage dynamique

```
public class chaine {
  private class cel { private char symb;
                      private cel suivant;
  }

  cel laChaine;

  public chaine() { laChaine=null;}

  public void ajoute(char c) {
     cel uneCel= new cel();
     uneCel.symb=c;
     uneCel.suivant=laChaine;
     laChaine=uneCel;
  }
}
```

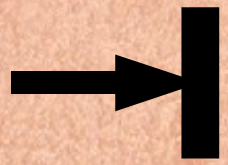
Structures de données chaînage dynamique

```
public class chaine {


   private class cel { private char symb;
                       private cel suivant;
   }
```



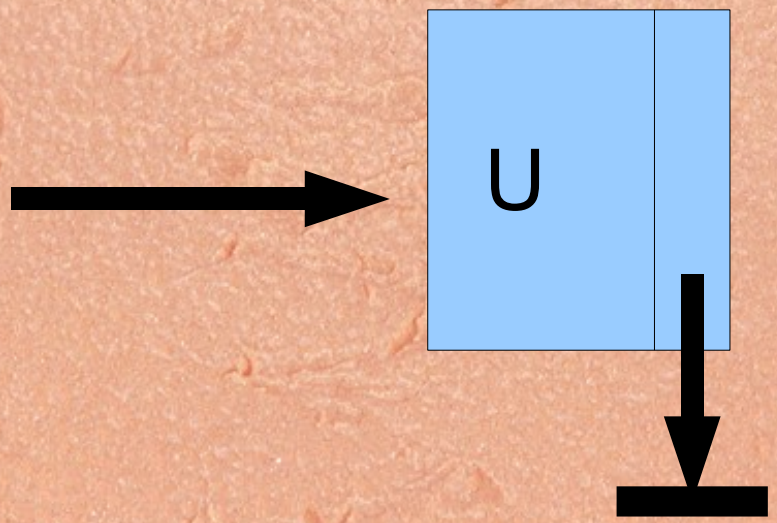
Structures de données chaînage dynamique

```
public class chaine {

  public chaine() { laChaine=null;}
```



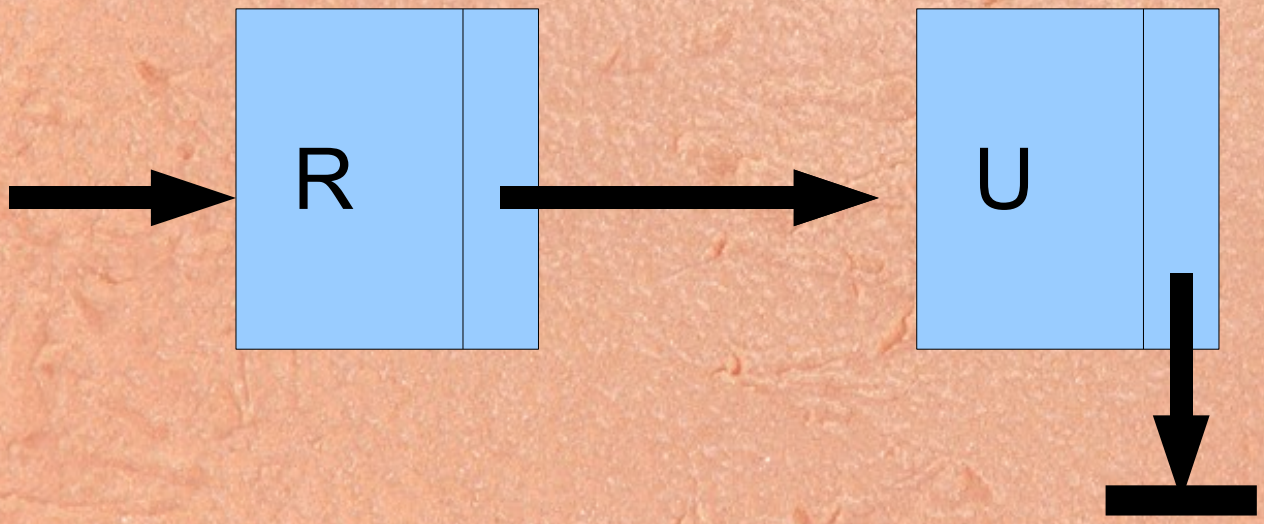
Structures de données chaînage dynamique

```
public class chaine {

   public void ajoute(char c) {
       cel uneCel= new cel();
       uneCel.symb=c;
       uneCel.suivant=laChaine;
       laChaine=uneCel;
   }
}
```



Structures de données chaînage dynamique

```
public class chaine {

   public void ajoute(char c) {
       cel uneCel= new cel();
       uneCel.symb=c;
       uneCel.suivant=laChaine;
       laChaine=uneCel;
   }
}
```



Structures de données généricité

```
public class couple <T> {
   T gauche;    T droite;

   public couple(T g, T d) {gauche=g;droite=d;}

   public T first() {
      return gauche;
   }

   public T last() {
      return droite;
   }

}
```