

Les templates génériques

- Un template générique est destinée à fournir le même type de fonctionnalités pour différents types de données
- On fournit alors des types de paramètres dans sa définition
 - `public void drawAll(HashSet<? extends Shape> shapes){ ... }`
 - `class Groupe<P> { ... }`
- On peut contraindre les types passés en paramètre :
`class NomClasse<P extends UneClasse & UneInterface & UneAutre> { ... }`
- On peut envoyer un type de classe en paramètre :
`boolean contains(HashSet<Personne> liste, Class<? extends Etudiant> type) {...}`

Intérêt de méthodes génériques

```
public static void drawAll(HashSet<Shape> shapes){  
    for(Shape shape : shapes) shape.draw();  
}  
  
public static void main(String[] args) {  
    HashSet<Shape> formes = new HashSet<>() ;  
    formes.add(new Circle(5)) ;  
    formes.add(new Circle(8)) ;  
    drawAll(formes);  
}
```

OK

```
public static void drawAll(HashSet<Shape> shapes){  
    for(Shape shape : shapes) shape.draw();  
}  
  
public static void main(String[] args) {  
    HashSet<Circle> formes = new HashSet<>() ;  
    formes.add(new Circle(5)) ;  
    formes.add(new Circle(8)) ;  
    drawAll(formes);  
}
```

ERREUR DE COMPILATION

```
public static void drawAll(HashSet<? extends Shape> shapes){  
    for(Shape shape : shapes) shape.draw();  
}  
  
public static void main(String[] args) {  
    HashSet<Circle> formes = new HashSet<>() ;  
    formes.add(new Circle(5)) ;  
    formes.add(new Circle(8)) ;  
    drawAll(formes);  
}
```

OK

Exemple de classe générique

```
public class Personne {
    private String prenom, nom ;

    public Personne(String prenom, String nom){
        this.prenom = prenom ;
        this.nom = nom ;
    }

    @Override
    public String toString() {
        return prenom + " " + nom.toUpperCase() ;
    }
}
```

```
public class Groupe<P extends Personne> {
    ArrayList<P> group ;

    public Groupe() {
        super() ;
        group = new ArrayList<P>() ;
    }

    public void ajouter(P personne){
        group.add(personne) ;
    }

    public P premier(){
        return group.get(0) ;
    }

    @Override
    public String toString(){
        String str = "--- Listing --- \n" ;
        for (P elem : group){
            str += elem + "\n" ;
        }
        return str ;
    }
}
```

```
public class Etudiant extends Personne {

    private int numCarte ;

    public Etudiant(int num, String prenom, String nom) {
        super(prenom, nom) ;
        numCarte = num ;
    }

    public int getNumCarte() {
        return numCarte ;
    }

    @Override
    public String toString() {
        return "Etudiant n° " + getNumCarte() + " -> " + super.toString();
    }
}
```

```
public static void main(String[] args) {
    Groupe<Etudiant> gr = new Groupe<Etudiant>() ;

    gr.ajouter(new Etudiant(987654321, "san", "goku")) ;
    gr.ajouter(new Etudiant(123456789, "tortue", "geniale")) ;

    System.out.println(gr);

    System.out.println("Le numéro du premier etudiant est "
        + gr.premier().getNumCarte() );
}
```

```
--- Listing ---
Etudiant n° 987654321 -> san GOKU
Etudiant n° 123456789 -> tortue GENIALE
```

```
Le numéro du premier etudiant est 987654321
```

Exemples de méthodes génériques

```
public static boolean contains(HashSet<? extends Personne> liste, Class<? extends Etudiant> type){
    for(Personne p : liste){
        if(type.isInstance(p)) return true ;
    }
    return false ;
}

public static <T extends Etudiant> HashSet<T> findAllIn(Class<T> typeEtudiant, HashSet<Personne> liste){
    HashSet<T> result = new HashSet<T>() ;
    for(Personne p : liste){
        if(typeEtudiant.isInstance(p)) result.add((T) p) ;
    }
    return result ;
}

public static void main(String[] args) {

    HashSet<Personne> personnes = new HashSet<>() ;
    personnes.add(new Personne("Cesar", "Bistruccla")) ;
    int i = 0 ;
    personnes.add(new EtudiantInfo(++i, "Terry", "Golo")) ;
    personnes.add(new EtudiantBio(++i, "Lara", "Tatouille")) ;
    personnes.add(new EtudiantInfo(++i, "Helmut", "Hardelpik")) ;

    boolean hasInfo = contains(personnes, EtudiantInfo.class) ;
    System.out.println("Il y a des etudiants en Info : " + hasInfo);

    HashSet<EtudiantBio> bios = findAllIn(EtudiantBio.class, personnes) ;
    System.out.println("\nEtudiants en Bio :");
    for(EtudiantBio etu: bios) System.out.println(etu);
}
```

Il y a des etudiants en Info : true

Etudiants en Bio :
Etudiant n° 2 -> Lara TATOUILLE