La Programmation Orientée Objet (POO) avec Java

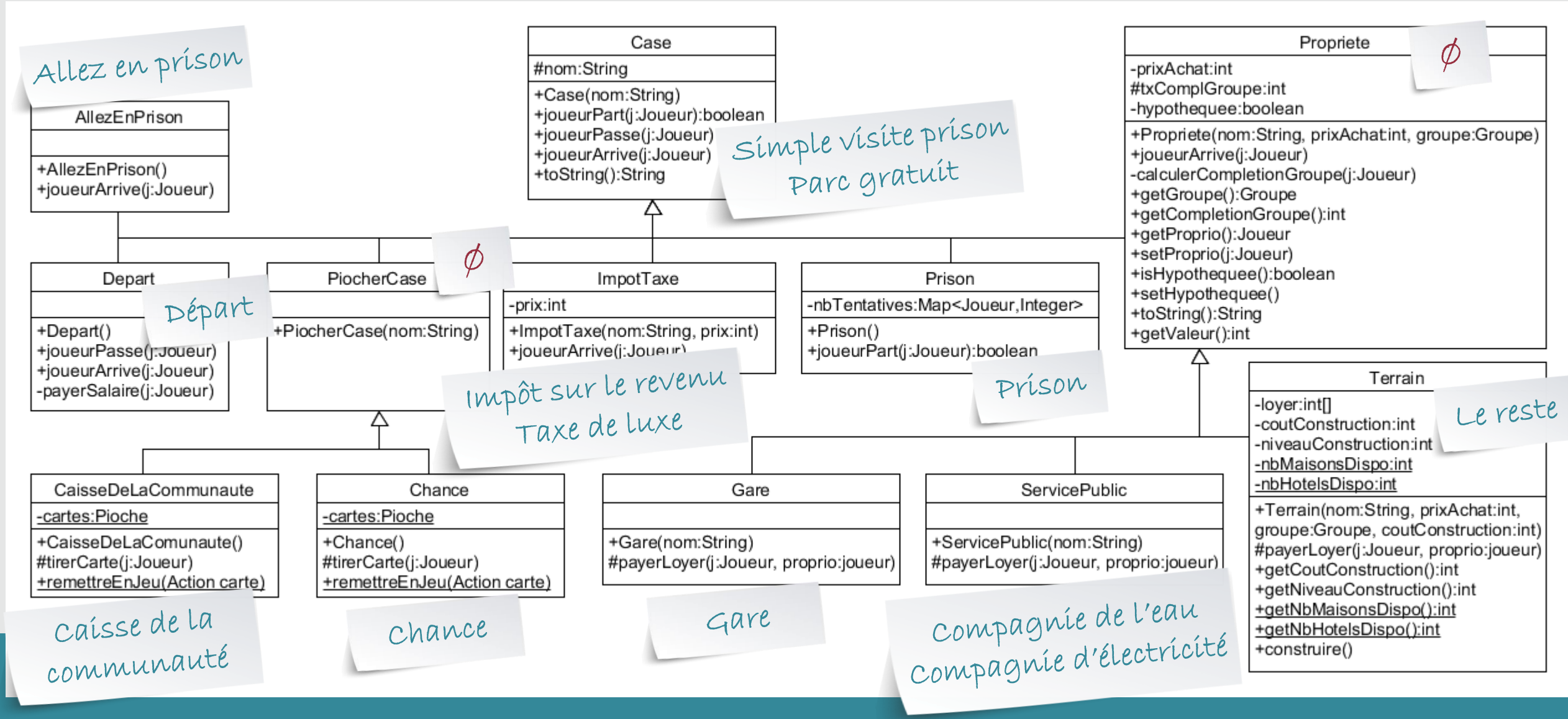
Module 6 – Les classes abstraites, les méthodes abstraites et les interfaces

Objectifs

- Découvrir la notion d'abstraction
- Comprendre la différence entre une classe concrète et une classe abstraite
- Savoir coder des classes abstraites contenant éventuellement des méthodes abstraites
- Comprendre la notion d'interface et son intérêt

Les classes abstraites, les méthodes abstraites et les interfaces

Les classes abstraites



Les classes abstraites

```
public abstract class Propriete extends Case {

  private int prixAchat;
  private Joueur proprio;
  private Groupe groupe;
  protected int txComplGroupe;
  private boolean hypothequee;

  public Propriete(String nom, int prixAchat, Groupe groupe) {
    super(nom);
      …
  }
      …
}
```

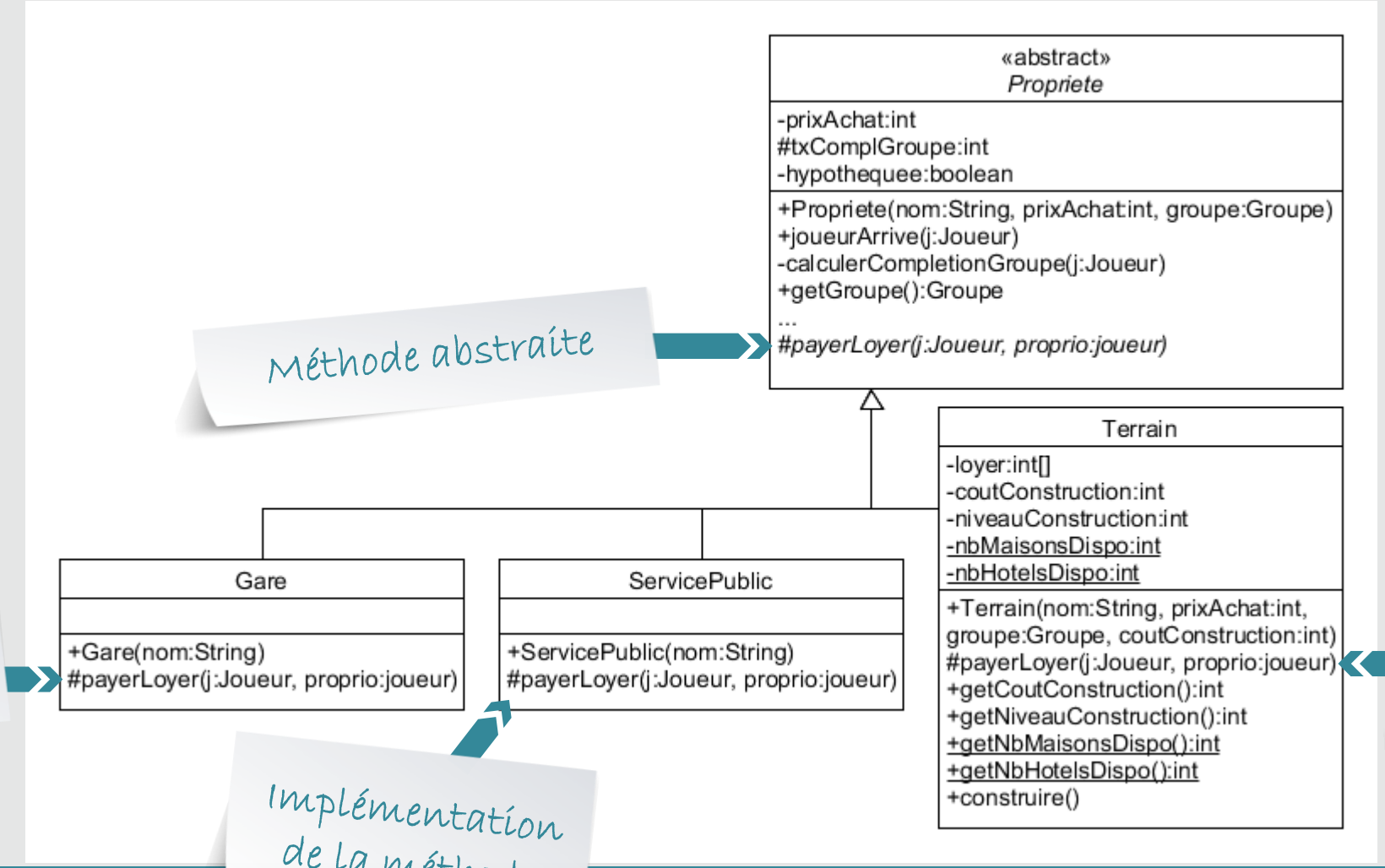
abstract permet d'empêcher l'instanciation d'une classe

Le compilateur interdit la construction d'une instance

```
Propriete p = new Propriete("Propriété", 200, Groupe.BLEU);
```

Les classes abstraites, les méthodes abstraites et les interfaces

Les méthodes abstraites



Les méthodes abstraites

```
public abstract class Propriete extends Case {
    …
  protected abstract void payerLoyer(Joueur j, Joueur proprio);
}

public class Terrain extends Propriete {
    …
  @Override
  protected void payerLoyer(Joueur locataire, Joueur proprietaire) {
    int loyer = this.loyers[this.niveauConstruction];
    if (this.niveauConstruction == 0 && this.txComplGroupe == 100)
      loyer *= 2;
    locataire.payeA(proprietaire, loyer);
  }
}
```

Les méthodes abstraites

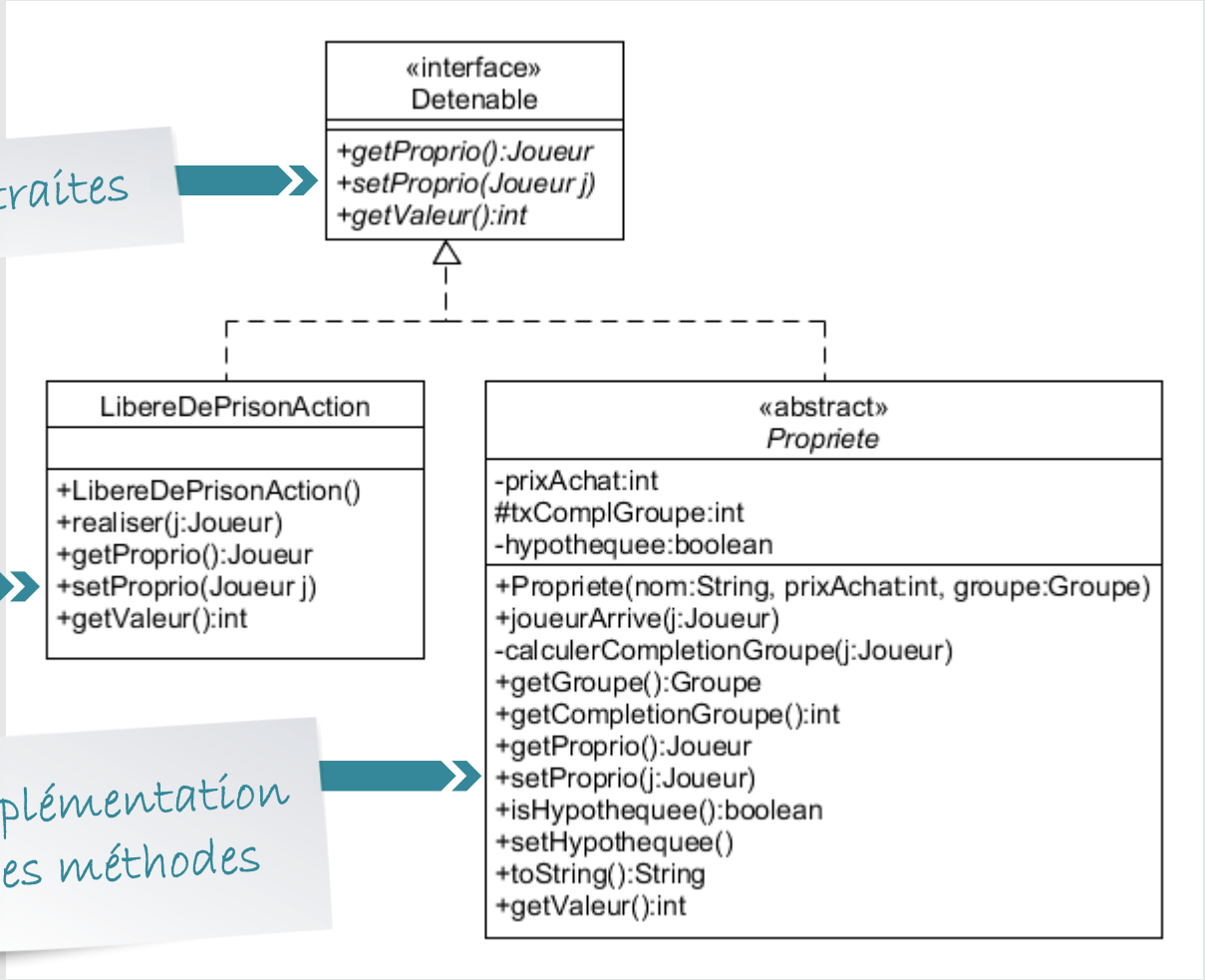
```
public class Gare extends Propriete {
    …
  @Override
  protected void payerLoyer(Joueur passager, Joueur p) {
    int loyer = this.txComplGroupe;
    if(loyer==75)
      loyer = 100;
    else if(loyer==100)
      loyer = 200;
    System.out.printf("%s possède %d gare%s%n", p, this.txComplGroupe*4/100, loyer>25?"s":"");
    passager.payeA(proprietaire, loyer);
  }
}
```

Les méthodes abstraites

```
public class ServicePublic extends Propriete {
    …
  @Override
  protected void payerLoyer(Joueur utilisateur, Joueur proprietaire) {
    int nb = this.txComplGroupe*2 /100;
    String s = nb<2?"":"s";
    int loyer = Monopoly.getDe1().getFaceTiree() + Monopoly.getDe2().getFaceTiree();
    if(nb==1) loyer *= 4;
    else loyer *= 10;
    System.out.printf("%s possède %d service%s public%s%n", proprietaire, nb, s, s);
    utilisateur.payeA(proprietaire, loyer);
  }
}
```

Les classes abstraites, les méthodes abstraites et les interfaces

Les interfaces



Les interfaces

```
public interface Detenable {
  Joueur getProprio();
  void setProprio(Joueur j);
  int getValeur();
}
```

Toutes les méthodes sont abstraites.
Le mot clef **abstract** n'est pas nécessaire.

Les interfaces

```
public class LibereDePrisonAction extends Action implements Detenable {
  private Joueur proprio;
  @Override
  public Joueur getProprio() {
    return this.proprio;
  }

  @Override
  public void setProprio(Joueur j) {
    this.proprio = j;
  }

  @Override
  public int getValeur() {
    return 50;
  }
   …
}
```

Les interfaces

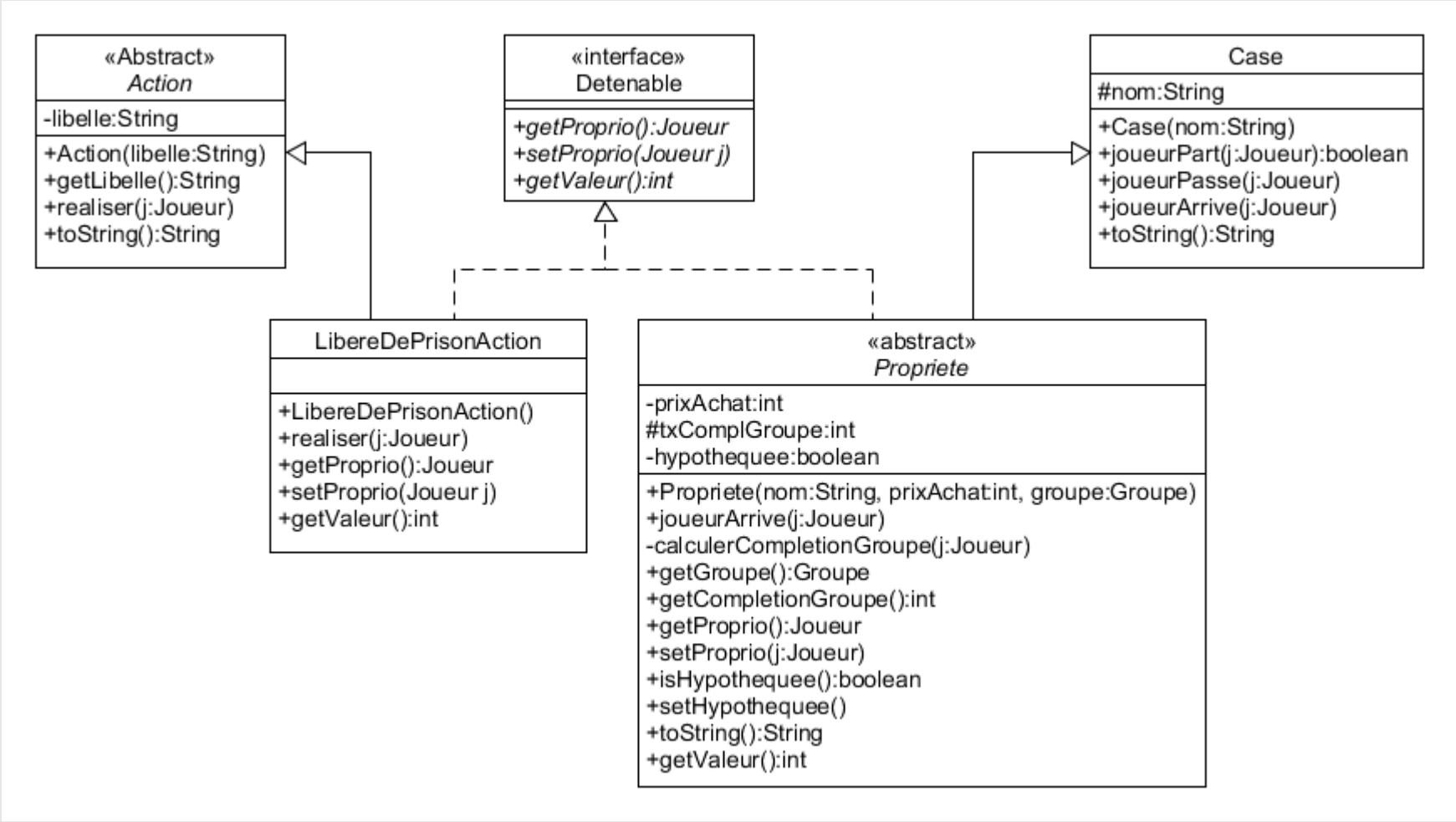
```
public abstract class Propriete extends Case implements Detenable {
    …
  private Joueur proprio;

  @Override
  public Joueur getProprio() {
    return this.proprio;
  }

  @Override
  public void setProprio(Joueur j) {
    Joueur ancienProprio = this.proprio;
    this.proprio = j;
    if(ancienProprio != null)
    this.calculerCompletionGroupe(ancienProprio);
    this.calculerCompletionGroupe(j);
  }
```

```
  @Override
  public int getValeur() {
    return this.prixAchat;
  }
   …
}
```

Les interfaces



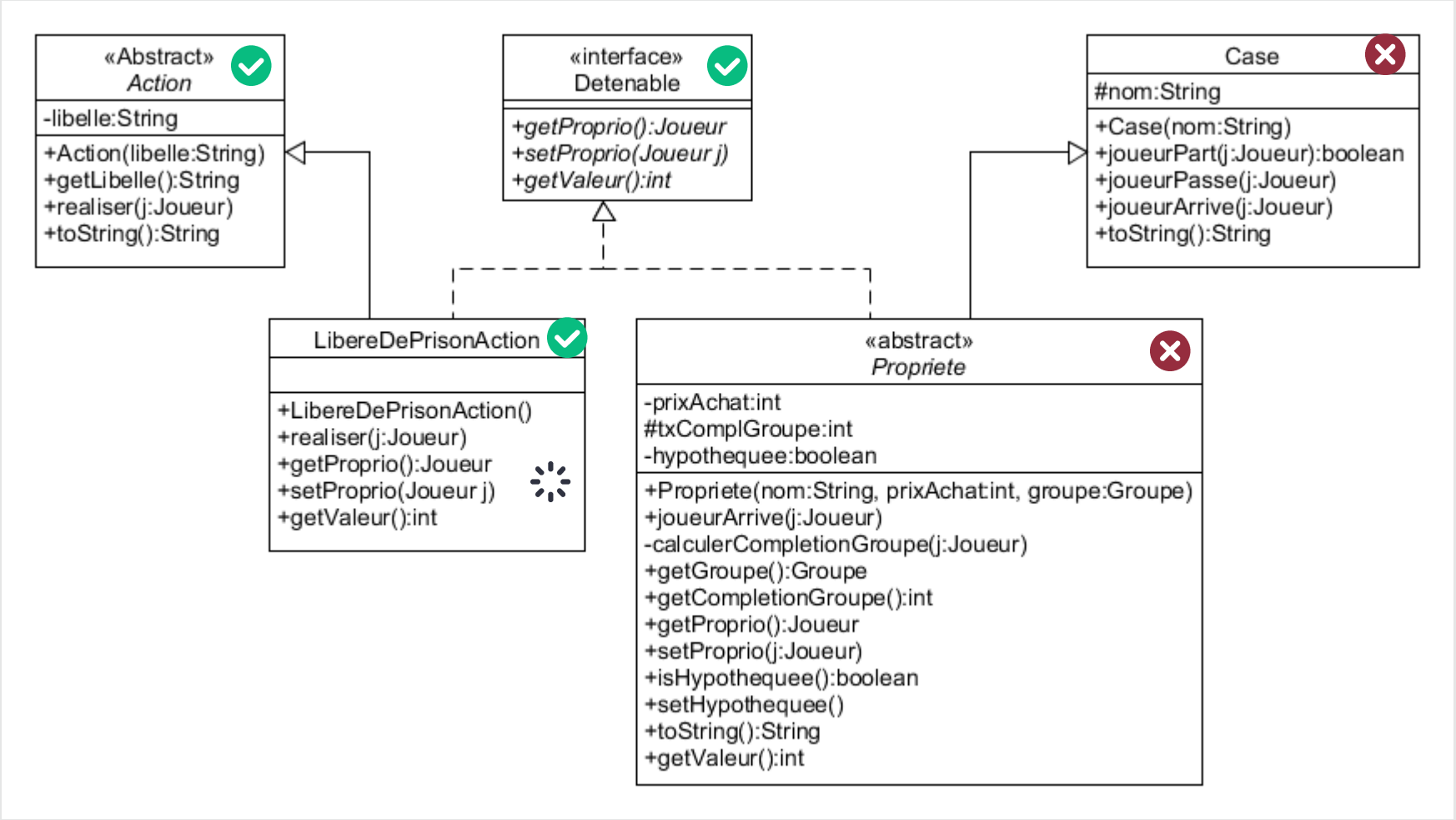
Bilan

Dans la définition de	Classe concrète	Classe abstraite	Interface
Déclaration	`class`	`abstract class`	`interface`
Attributs	✓	✓	✗
Constantes	✓	✓	✓
Constructeur	✓	✓	✗
Méthode concrète	✓	✓	✗
Méthode abstraite (mot clef)	✗	✓ (`abstract`)	✓ (Ø)

Dans l'utilisation de	Classe concrète	Classe abstraite	Interface
Déclaration	`extends`	`extends`	`implements`
Minimum	0	0	0
Maximum	1	1	∞

Les classes abstraites, les méthodes abstraites et les interfaces

Le polymorphisme et les interfaces



Les classes abstraites, les méthodes abstraites et les interfaces

Le polymorphisme et les interfaces

- Exemple : pour procéder aux échanges de cartes entre joueurs
  - Deux types de carte échangeables :
    - Titre de propriété
    - Cartes Chance ou Caisse de la Communauté « Vous êtes libéré de prison »
  - Les deux classes modélisant ces cartes implémentent l'interface « Detenable »
- La méthode **`getCartesEchangeables()`** retourne un ensemble d'éléments Detenable

TP