Paradigme Objet

- Représentation
  - Abstraction d’un système
- Encapsulation
  - Délégation de la responsabilité à l’objet de ses traitements
- Polymorphisme
  - Capacité d’un objet à appartenir à plusieurs types
- Héritage
  - Principe de classe parent-enfant

Héritage

- Augmentation d’un objet de base abstrait
- Ajout de spécifications
- Ré-écriture de méthode (Polymorphisme)
- Exemple classe ‘Vehicule’, ‘Voiture’, ‘Moto’
- Pas d’héritage multiple => utilisation des interfaces

Classe JAVA

```java
public class HelloWorld {

  public static void main(String[] args) {
    System.out.println("HelloWorld !");
  }

}
```

Classe JAVA #2

```java
public class AddMachine {

  private int attribute1;
  private int attribute2;

  public AddMachine(int attribute1, int attribute2) {
    this.attribute1 = attribute1;
    this.attribute2 = attribute2;
  }

  public String getTotal() {
    Integer total = attribute1 + attribute2;
    return total.toString();
  }

  public static void main(String[] args) {
    AddMachine hw = new AddMachine(1, 2);
    System.out.println(hw.getTotal());
  }
}
```

Les scopes

Access Levels

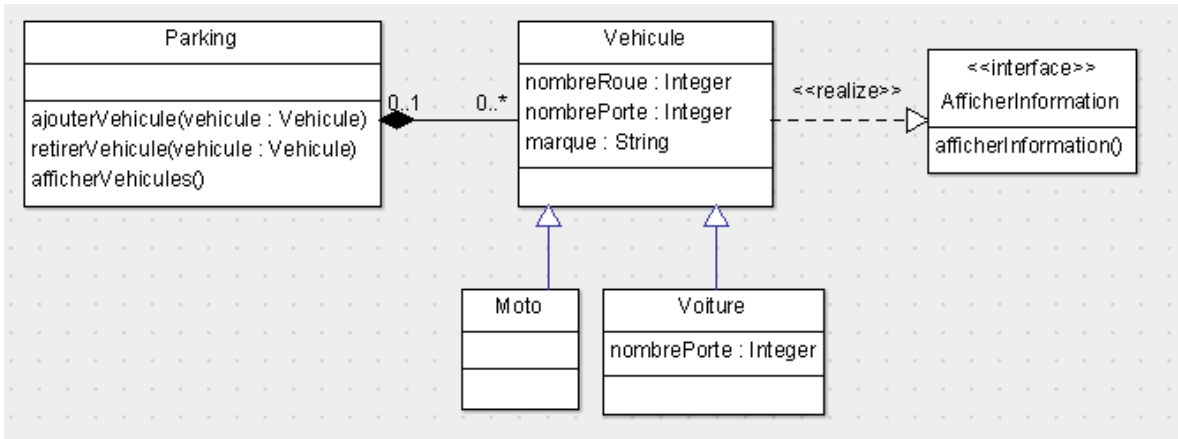
Modifier	Class	Package	Subclass	World
`public`	Y	Y	Y	Y
`protected`	Y	Y	Y	N
no modifier	Y	Y	N	N
`private`	Y	N	N	N

Mots clés Java

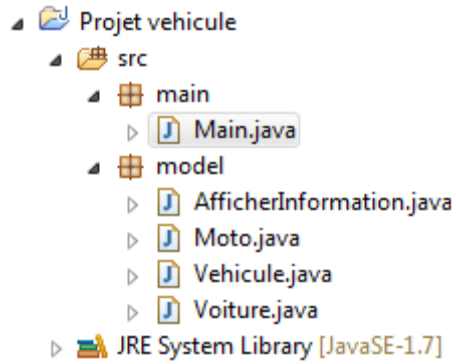
- Entités:
  - *Package, class, interface, enum*
- ‘Modifier’:
  - *public, protected, private*
- Héritage, implémentation:
  - *abstract, extends, implements*
- Autres:
  - *import*
  - *final, static*
  - *synchronised*
  - *super*
  - *void, this*

Exemple:

- Diagramme UML de classe



- Architecture du projet



Exemple: Interface AfficherInformation

- Toutes les méthodes sont publiques
- Pas de corps de méthode
- Polymorphisme

```
1  package model;
2
3  public interface AfficherInformation {
4
5      public void afficherInformation();
6
7  }
```

Exemple: Classe Vehicule

```java
public abstract class Vehicule implements AfficherInformation{

    protected int nombreRoue;
    protected String marque;

    public Vehicule(int nombreRoue, String marque) {
        super();
        this.nombreRoue = nombreRoue;
        this.marque = marque;
    }

    @Override
    public void afficherInformation() {
        System.out.println("Vehicule [nombreRoue=" + nombreRoue + ", marque=" + marque
                + "]");
    }

    public int getNombreRoue() {
        return nombreRoue;
    }

    public void setNombreRoue(int nombreRoue) {
        this.nombreRoue = nombreRoue;
    }

    public String getMarque() {
        return marque;
    }

    public void setMarque(String marque) {
        this.marque = marque;
    }

}
```

Exemple: Classe Voiture

```java
public class Voiture extends Vehicule{

    protected int nombrePorte;

    public Voiture(int nombreRoue, int nombrePorte, String marque) {
        super(nombreRoue, marque);
        this.nombrePorte = nombrePorte;
    }

    @Override
    public void afficherInformation() {
        System.out.println("Voiture [nombreRoue=" + nombreRoue + ", nombrePorte=" + nombrePorte + ", marque=" + marque + "]");
    }

    public int getNombrePorte() {
        return nombrePorte;
    }

    public void setNombrePorte(int nombrePorte) {
        this.nombrePorte = nombrePorte;
    }

}
```

Exemple: Classe Moto

```java
public class Moto extends Vehicule{

    public Moto(int nombreRoue, String marque) {
        super(nombreRoue, marque);
    }

    @Override
    public void afficherInformation() {
        System.out.println("Moto [nombreRoue=" + nombreRoue + ", marque=" + marque + "]");
    }

}
```

Exemple: Classe Parking

```java
import java.util.ArrayList;
import java.util.List;

public class Parking {

    private List<Vehicule> vehicules = new ArrayList<Vehicule>();

    public boolean ajouterVehicule(Vehicule vehicule){
        return vehicules.add(vehicule);
    }

    public boolean retirerVehicule(Vehicule vehicule){
        return vehicules.remove(vehicule);
    }

    public void afficherVehicule(){
        if (vehicules != null && vehicules.size() > 0){
            for (Vehicule vehicule : vehicules){
                vehicule.afficherInformation();
            }
        }
    }

}
```

Exemple: Classe Main

```java
package main;

import model.Moto;
import model.Parking;
import model.Voiture;

public class Main {

    public static void main(String[] args) {
        Parking parking = new Parking();
        Voiture mercedes = new Voiture(4, 5, "Mercedes");
        Moto yamaha = new Moto(2, "Yamaha");
        Voiture lada = new Voiture(4, 5, "Lada");

        parking.ajouterVehicule(mercedes);
        parking.ajouterVehicule(yamaha);
        parking.ajouterVehicule(lada);

        parking.afficherVehicule();
    }

}
```

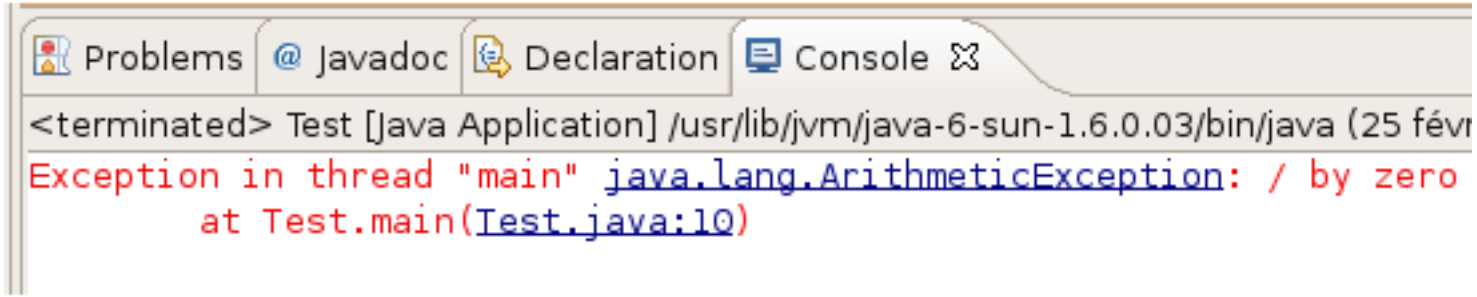
```
Voiture [nombreRoue=4, nombrePorte=5, marque=Mercedes]
Moto [nombreRoue=2, marque=Yamaha]
Voiture [nombreRoue=4, nombrePorte=5, marque=Lada]
```

Methodes spéciales

- public boolean equals(Object o)
- public String toString()
- public int hashCode()

Les exceptions

```java
public static void main(String[] args) {

  int j = 20, i = 0;
  try {
    System.out.println(j/i);
  } catch (ArithmeticException e) {
    System.out.println("Division par zéro !");
    System.out.println(e.getMessage());
  }
  System.out.println("ouf !");
}
```



Les exceptions 2

```
class NombreHabitantException extends Exception{
  public NombreHabitantException(){
    System.out.println("Vous essayez d'instancier une classe Ville avec un nombre d'habitants négatif !");
  }
}
```

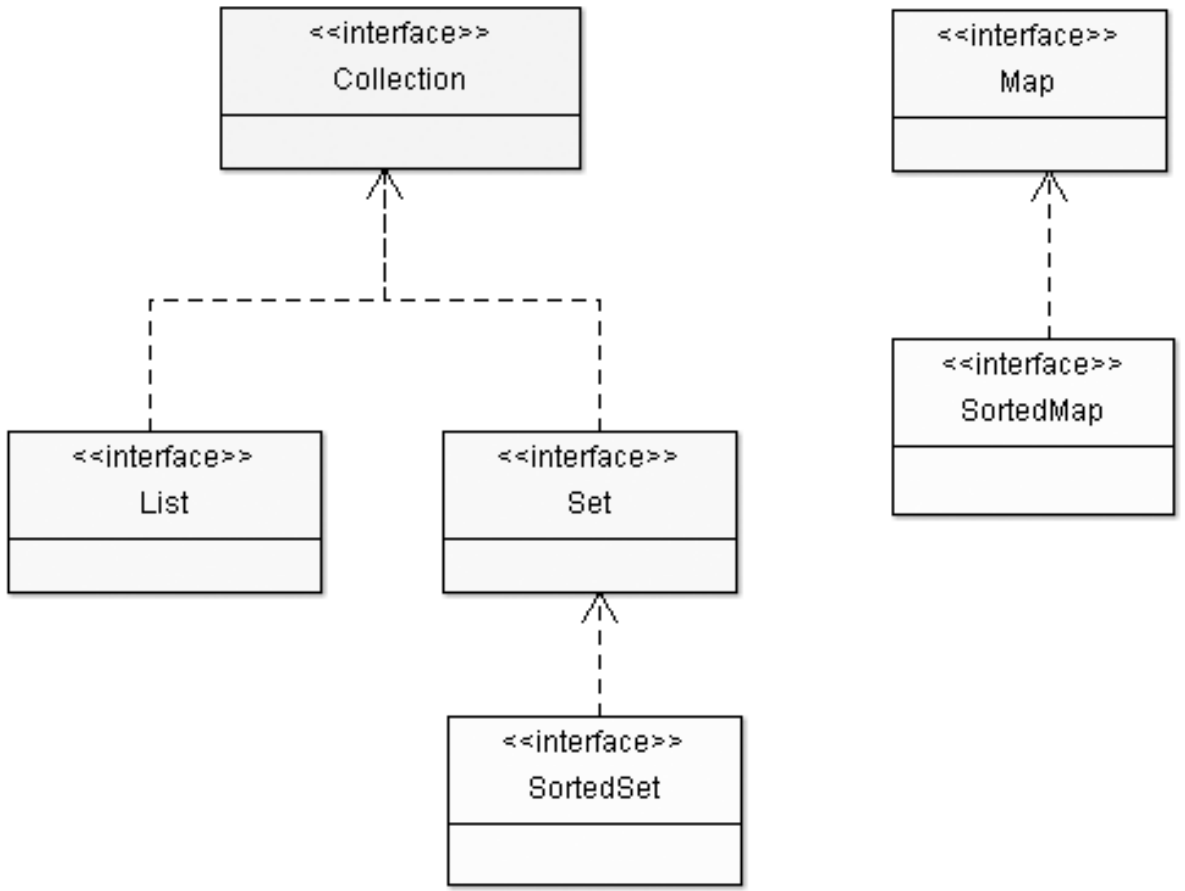
```
public Ville(String pNom, int pNbre, String pPays)
  throws  NombreHabitantException
  {
    if(pNbre < 0)
      throw new NombreHabitantException();
    else
    {
      nbreInstance++;
      nbreInstanceBis++;

      nomVille = pNom;
      nomPays = pPays;
      nbreHabitant = pNbre;
      this.setCategorie();
    }
  }
```

Les enumérations

```
public enum Langage {
  JAVA,
  C,
  CPlus,
  PHP;
}
```

```
public class Main {
  public static void main(String args[]){
    for(Langage lang : Langage.values()){
      if(Langage.JAVA.equals(lang))
        System.out.println("J'aime le : " + lang);
      else
        System.out.println(lang);
    }
  }
}
```

Les collections

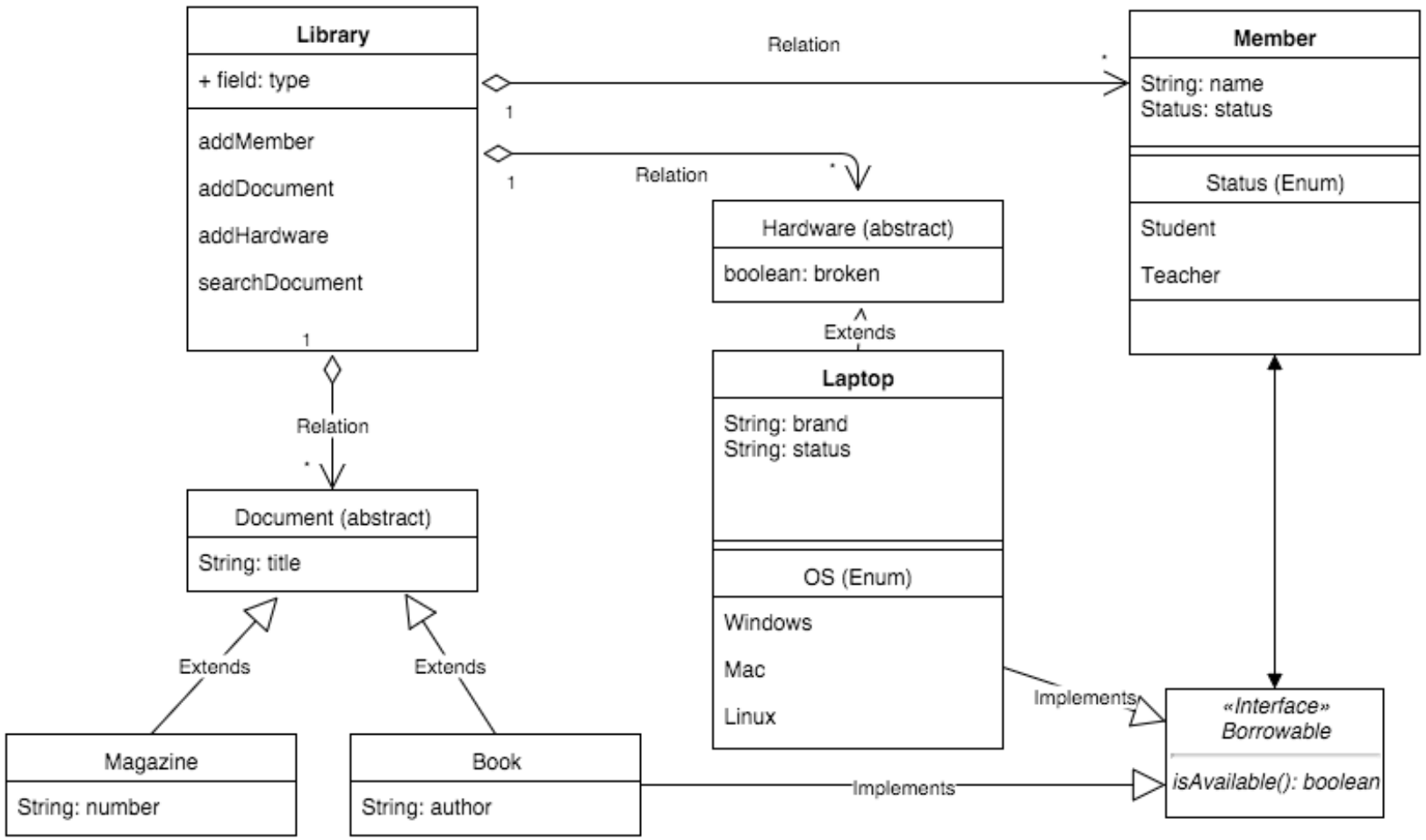


Eclipse

- http://www.eclipse.org/downloads/packages/release/Neon/M4A

- Eclipse Neon, IDE for Java Developers

TP #2


Library
+ field: type
addMember
addDocument
addHardware
searchDocument
Relation
1
*
Member
String: name
Status: status
Status (Enum)
Student
Teacher
1
Relation
*
Hardware (abstract)
boolean: broken
Extends
Laptop
String: brand
String: status
OS (Enum)
Windows
Mac
Linux
1
Relation
*
Document (abstract)
String: title
Extends
Extends
Magazine
String: number
Book
String: author
Implements
Implements
«Interface»
Borrowable
isAvailable(): boolean

TP #2

1 – Implémenter le modèle objet et le peupler

1 – Implémenter la méthode showBorrowed pour afficher les documents et le matériel empruntés par un membre

1 – Implémenter la méthode pour rechercher un document avec un mot clé

1 - Implémenter la méthode pour emprunter un document

1 – Lever une exception lorsque un membre emprunte plus de 5 éléments