lab-sticc.univ-brest.fr/~babau/

Exemple de code et modèles UML

Application au Design Pattern Delegator

Jean-Philippe Babau

Département Informatique, UFR Sciences, UBO
Laboratoire Lab-STICC

Design pattern : Delegator

```
public class Personne {

    protected String nom;
    protected int age;
    protected PersonnePrinter lePrinter;

    public Personne() {
        nom = "pierre";     age = 30;
        lePrinter = new PersonnePrinter();
    }

    public Personne(String unNom, int unAge) {
        nom = unNom;        age = unAge;
        lePrinter = new PersonnePrinter();
    }

    public void Anniversaire () {age += 1;}

    public void Print() {lePrinter.Print(this);}

    public String Nom() {return nom;}

    public int Age() {  return age; }
}
```

Design pattern : Delegator

```
public class PersonnePrinter {

    public void Print(Personne unePersonne) {
        System.out.print("personne : "+unePersonne.Nom());
        if (unePersonne.Age() >= 18) {
            System.out.println(" majeur");
        } else {
            System.out.println(" mineur");
        }
    }
}
```

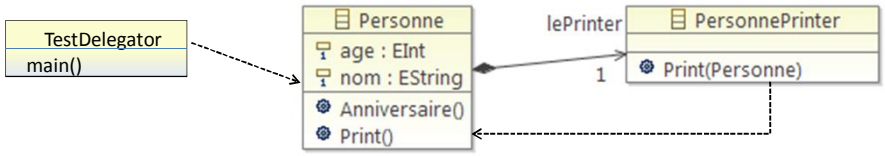
Tests et exécution

```
public class testDelegator {

    public static void main(String[] args) {

        Personne pierre = new Personne();
        pierre.Print();

        Personne paul = new Personne("Paul",17);
        paul.Print();
        paul.Anniversaire();
        paul.Print();

    }
}
```

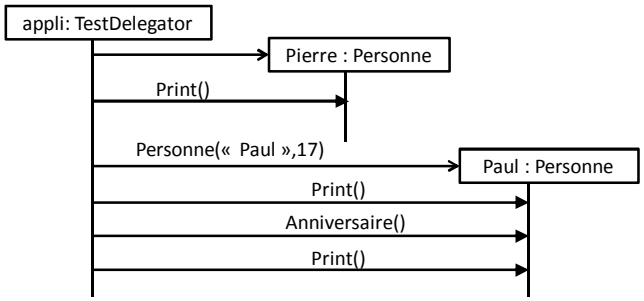
Problems @ Javadoc Declaration Console Synchronize

```
<terminated> testDelegator [Java Application] C:\Program Files\Java\jre6\bin\ja
personne : Pierre majeur
personne : Paul mineur
personne : Paul majeur
```

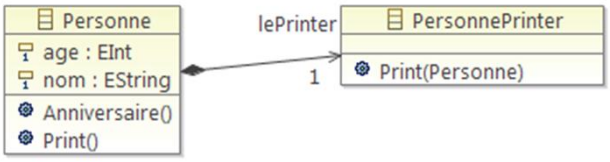
UML

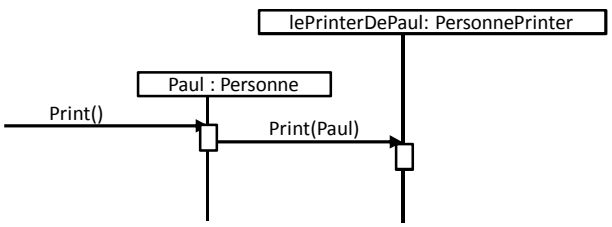


Exécution de main()



UML





Analyse

- PersonnePrinter devrait être un singleton
  - Il n'est pas utile de créer un objet pour chaque objet Personne
- Afin de rendre la classe PersonnePrinter indépendante de l'implémentation de Personne, les paramètres de la méthode Print() peuvent être modifiés

```
public class PersonnePrinter {

    public void Print(String unNom, int unAge) {
        System.out.print("personne : "+unNom);
        if (unAge >= 18) {
            System.out.println(" majeur");
        } else {
            System.out.println(" mineur");
        }
    }
}
```

- Il devrait y avoir des contrôles de type dans les constructeurs de Personne
  - **age** est positif
  - **nom** ne contient que des lettres

Conclusion

- Les diagrammes UML doivent être enrichis de commentaires
  - Pour décrire les contraintes sur les classes (invariants)
  - Pour décrire le comportement des méthodes
- Exercices
  - Passer de Java à UML
    - Décrire les classes java aux travers de diagrammes de classe UML
      - plusieurs diagrammes UML
    - Illustrer des exécutions au travers de diagrammes de séquence
      - plusieurs exécutions
      - plusieurs décompositions
  - Passer de UML à Java : coder une solution en respectant les contraintes exprimées par les schémas
    - Les schémas et les contraintes doivent être complets