Génie logiciel avancé

Spécification, Conception, Développement et Gestion du cycle de vie d'une solution informatique

2015-2016

université PARIS DIDEROT PARIS 7

Plan des cours

1. Modalités du cours,
   Présentation du **génie logiciel**,
   Cycles de développement,
2. Cycles de développement (suite),
   Spécifications
3. Planification de projet,
   Conception, Jersey
4. Suivi de projet, UML, DataNucleus
5. Design patterns
6. Reflexion, Annotations
7. Environnement de développement :
   IDE, SCM, Environnement de test, Intégration continue, tests.
8. Revues de code, refactoring
9. Recettes, présentation client
10. Déploiement,
    Gestion des évolutions fonctionnelles
11. Présentation des projets par les étudiants
12. Séance de questions / réponses

Revue de code

PROs

Détection de bug

Vérification des règles de développement

Amélioration globale de la qualité du code

CONs

Activité complètement subjective

Travail sans fin

Ne peut se faire qu'avec le code d'un tiers

Nécessite de l'expérience

Revue de code – Niveaux d’analyse

1. Syntaxique :
   - Nommage (classes, fonctions, variables)
   - Commentaires de code

2. Bas-niveau :
   - Code mort (fonctions inutilisées)
   - Code inutile (debug, etc.)
   - Manque de logique d’implémentation
   - Anti-pattern

3. Haut-niveau
   - Manque de logique de structure
   - Plat de nouilles

Revue de code – Niveaux d'analyse

1. Syntaxique :
   - Nommage (classes, fonctions, variables)
   - Commentaires de code

   → Partiellement automatique - Checkstyle

2. Bas-niveau :
   - Code mort (fonctions inutilisées)
   - Code inutile (debug, etc.)
   - Manque de logique d'implémentation
   - Anti-pattern

   → Partiellement automatique - SonarQube

3. Haut-niveau
   - Manque de logique de structure
   - Plat de nouilles

   → Manuel

Exemple 1

```
public class DateParser {
        public static Date StringToDate(String date) {
                SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");

                try {

                        Date newdate = formatter.parse(date);
                        return newdate;

                } catch (ParseException e) {
                        e.printStackTrace();
                }
                return null;
        }

        public static void main(String[] args) {
                System.out.println(StringToDate("10/6/2000"));
        }
}
```

Exemple 1 - Syntaxique

Majuscule réservée pour les constructeurs

```
public class DateParser {
        public static Date StringToDate(String date) {
                SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");

                try {

                        Date newdate = formatter.parse(date);
                        return newdate;

                } catch (ParseException e) {
                        e.printStackTrace();
                }
                return null;
        }

        public static void main(String[] args) {
                System.out.println(StringToDate("10/6/2000"));
        }
}
```

Retour à la ligne inutile

Exemple 1 – Bas-niveau

```
public class DateParser {
        public static Date StringToDate(String date) {
                SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");

                try {

                        Date newdate = formatter.parse(date);
                        return newdate;

                } catch (ParseException e) {
                        e.printStackTrace();
                }
                return null;
        }

        public static void main(String[] args) {
                System.out.println(StringToDate("10/6/2000"));
        }
}
```

Absence de logger

Présence d'un « main »

Exemple 1 – Haut-niveau

```
public class DateParser {
        public static Date StringToDate(String date) {
                SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");

                try {

                        Date newdate = formatter.parse(date);
                        return newdate;

                } catch (ParseException e) {
                        e.printStackTrace();
                }
                return null;
        }

        public static void main(String[] args) {
                System.out.println(StringToDate("10/6/2000"));
        }
}
```

Format de date non standard

Exemple 2

```
public class DAOFactory {

	static final private PersistenceManagerFactory pmf=JDOHelper.getPersistenceManagerFactory("FlightPlanificationSystemDataBase");

	static final private AirportDAO airportDAO=new AirportDAOImpl(DAOFactory.pmf);
	static final private PlaneDAO planeDAO=new PlaneDAOImpl(DAOFactory.pmf);
	static final private UserDAO userDAO=new UserDAOImpl(DAOFactory.pmf);
	static final private CrewDAO crewDAO=new CrewDAOImpl(DAOFactory.pmf);
	static final private FlightDAO flightDAO=new FlightDAOImpl(DAOFactory.pmf);

	public static AirportDAO getAirportDAO(){
		return DAOFactory.airportDAO;
	}

	public static PlaneDAO getPlaneDAO(){
		return DAOFactory.planeDAO;
	}

	public static UserDAO getUserDAO(){
		return DAOFactory.userDAO;
	}
```

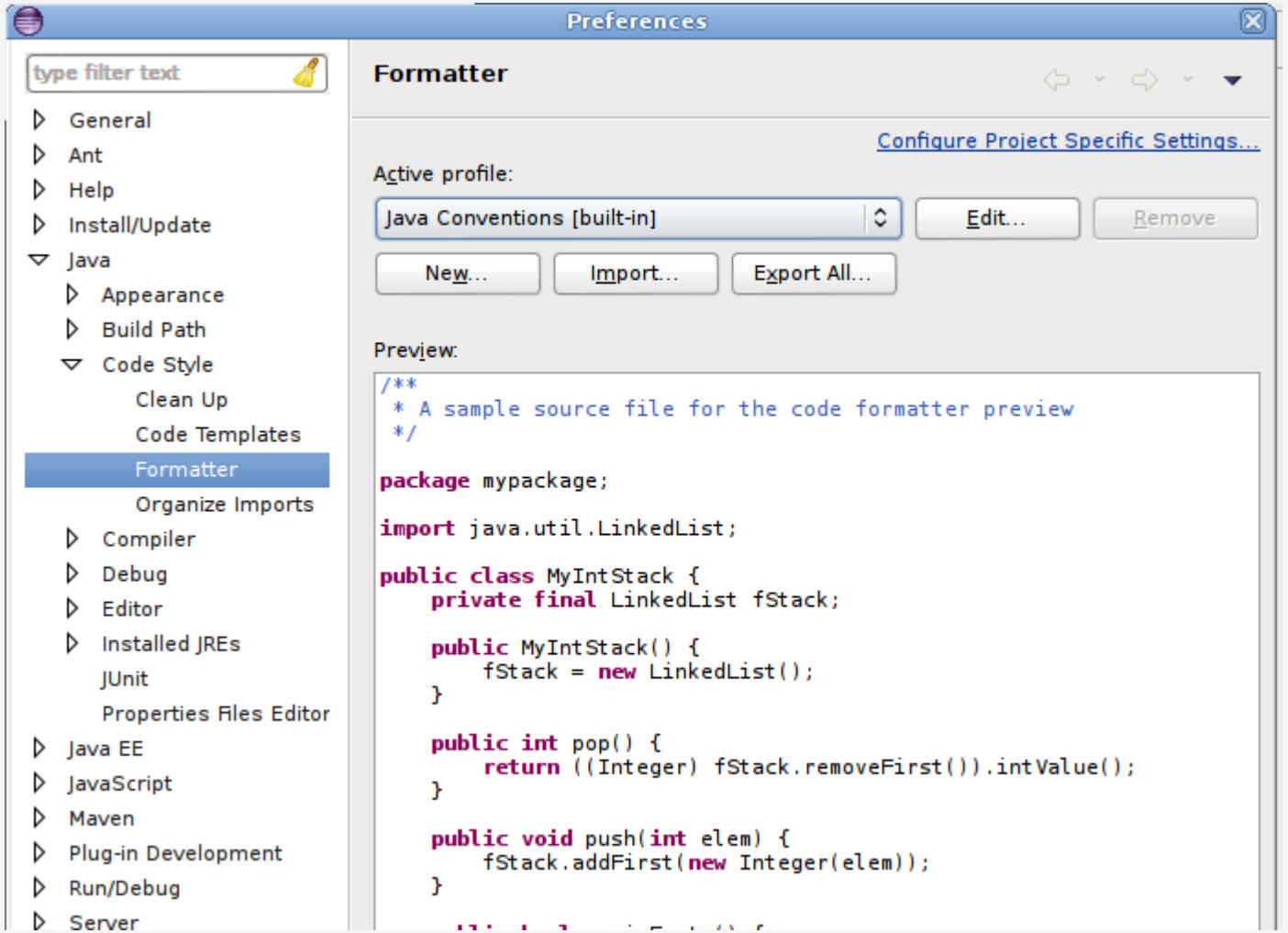
Revue de code – Syntaxique

- Conventions de nommage, par exemple :

Nom	Convention
nom de la classe	devrait commencer par lettre majuscule et être un nom par exemple String, en couleur, Button, Système, Thread etc.
nom de l'interface	devrait commencer par lettre majuscule et être un adjectif, par exemple Runnable, à distance, ActionListener etc.
nom de la méthode	devrait commencer par lettre minuscule et être un verbe, par exemple actionPerformed (), main (), print (), println (), etc.
nom de la variable	devrait commencer par lettre minuscule exemple firstName, orderNumber etc.
nom du paquet	devrait être en lettre minuscule, par exemple Java, lang, sql, util etc.
constantes nom	devrait être en lettre majuscule. par exemple, rouge, jaune, MAX_PRIORITY etc.

Revue de code – Syntaxique

- Format du code (indentation, retour à la ligne, etc.)



Revue de code – Bas-niveau

- Fonctionnement local de l'application, quelques exemples :
  - Utilisation d'un système de logging
  - Gestion des exceptions
  - Gestion des ressources (fichiers, sockets, etc.)
  - Structuration des codes (main, debug, tests)
  - Déclaration des constantes

Revue de code – Haut-niveau

- Fonctionnement global de l'application
- Détection d'une structure ou d'un comportement qui peut amener à :
  - Une incompréhension pour des développements futurs
  - Une absence de factorisation du code
  - Une mauvaise séparation du code
- Peut amener à des corrections… pas de manière systématique

Exemple 3 – Séparation des rôles

```
@PUT
@Consumes(MediaType.MULTIPART_FORM_DATA)
@Path("/import")
public void importFlights(@FormDataParam("file") InputStream uploadedInputStream) {
        Flight flight;
        String id, numCom, numAtc, oaciDep, oaciArr;
        Date depDate = null, arrDate = null;

        try {
                POIFSFileSystem fs = new POIFSFileSystem(uploadedInputStream);
                HSSFWorkbook wb = new HSSFWorkbook(fs);
                HSSFSheet sheet = wb.getSheetAt(0);

                for(Row row : sheet) {
                        try {
                                numCom = row.getCell(0).getStringCellValue();
                                numAtc = row.getCell(1).getStringCellValue();
                                oaciDep = row.getCell(2).getStringCellValue().substring(0, 4);
                                oaciArr = row.getCell(3).getStringCellValue().substring(0, 4);
                                depDate = row.getCell(4).getDateCellValue();
                                arrDate = row.getCell(5).getDateCellValue();
```

Exemple 3 – Séparation des rôles

```
@PUT
@Consumes(MediaType.MULTIPART_FORM_DATA)
@Path("/import")
public void importFlights(@FormDataParam("file") InputStream uploadedInputStream) {
        Flight flight;
        String id, numCom, numAtc, oaciDep, oaciArr;
        Date depDate = null, arrDate = null;

        try {
                POIFSFileSystem fs = new POIFSFileSystem(uploadedInputStream);
                HSSFWorkbook wb = new HSSFWorkbook(fs);
                HSSFSheet sheet = wb.getSheetAt(0);

                for(Row row : sheet) {
                        try {
                                numCom = row.getCell(0).getStringCellValue();
                                numAtc = row.getCell(1).getStringCellValue();
                                oaciDep = row.getCell(2).getStringCellValue().substring(0, 4);
                                oaciArr = row.getCell(3).getStringCellValue().substring(0, 4);
                                depDate = row.getCell(4).getDateCellValue();
                                arrDate = row.getCell(5).getDateCellValue();
```

Web-service
Chargement d'un fichier Excel