

### Exercise 01: (3)

Which of the following constraints can be used in a Linear Program: (with justification)

- 1)  $e^{-x_1} + x_2 \leq 12$  ; 2)  $x_1 + |5x_2| x_3 = 0,5$  ; 3)  $|5x_1 + x_2| = + 0.666$  ; 4)  $-x_1 - x_2 \leq -5/3$  ;
- 5)  $\sum_{j=1}^N x_{ij} x_j \leq b_i$  ; 6)  $-x_3 - \ln(x_2) \leq -5/3$

### Exercise 02: (6)

A) Determine all the basic solutions for the following equation system and specify for each solution

is feasible or not : 
$$\begin{cases} x_1 + x_2 + x_3 + 2x_4 \geq 3 \\ -x_2 + x_3 + 2x_4 = 2 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

B) Write the Canonical form (Max) of this LP:

$$\text{MIN } Z = 3x_1 - 3x_2 + 7x_3 \begin{cases} x_1 + 9x_2 - 7x_3 \geq 5 \\ |5x_1 + x_2| = 3 \\ 1 \leq x_1 + x_2 - 7x_3 \leq 5 \\ x_1, x_2 \geq 0 \end{cases}$$

### Exercise 03 : (3)

let consider the following LP:  $\text{MIN } Z = 3x_1 - 3x_2 + 7x_3 + 2x_4 \begin{cases} x_1 + x_2 + x_3 - 3x_4 \geq 2 \\ -x_1 + x_3 + x_4 = 7 \\ x_1 + 4x_2 - 7x_3 \geq 3 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$

A) Write the corresponding LP for Phase I and Phase II.

### Exercise 04: (8)

Soit le (PL) suivant :  $\text{Min } z = -5x_1 + 6x_2 \begin{cases} -x_1 + 2x_2 \leq 5 \\ 5x_1 + 3x_2 = 6 \\ x_1 + 2x_2 \geq 3 \\ x_2 \geq 0 \\ x_1 \text{ unrestricted in sign} \end{cases}$

- 1) A). Write the standard form of linear program to solve by simplex. B). Why do we need to add two artificial variables? C). What are the main variables and the slack variables?
- 2) A). Look for a realizable initial base solution. B). What are the base variables and non-base variables? C). Write the initial simplex table T0. D). Find the pivot.
- 3) A). Give the 2nd table.
- 4) A). Give the sequence of simplex tables and the optimal solution of this linear program.

**Exercise 01: (3)**

Which of the following constraints can be used in a Linear Program: (with justification)

- 1)  $e^{-x_1} + x_2 \leq 12$  ; 2)  $x_1 + |5x_2| x_3 = 0,5$  ; 3)  $|5x_1 + x_2| = + 0.666$  ; 4)  $-x_1 - x_2 \leq -5/3$  ;
- 5)  $\sum_{j=1}^N x_{ij} x_j \leq b_i$  ; 6)  $-x_3 - \ln(x_2) \leq -5/3$

**Exercise 02: (6)**

A) Determine all the basic solutions for the following equation system and specify for each solution

is feasible or not ?:

$$\begin{cases} x_1 + x_2 + x_3 + 2x_4 \geq 3 \\ -x_2 + x_3 + 2x_4 = 2 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

B) Write the Canonical form (Max) of this LP:

$$\text{MIN } Z = 3x_1 - 3x_2 + 7x_3 \begin{cases} x_1 + 9x_2 - 7x_3 \geq 5 \\ |5x_1 + x_2| = 3 \\ 1 \leq x_1 + x_2 - 7x_3 \leq 5 \\ x_1, x_2 \geq 0 \end{cases}$$

**Exercise 03 : (3)**

let consider the following LP:  $\text{MIN } Z = 3x_1 - 3x_2 + 7x_3 + 2x_4$

$$\begin{cases} x_1 + x_2 + x_3 - 3x_4 \geq 2 \\ -x_1 + x_3 + x_4 = 7 \\ x_1 + 4x_2 - 7x_3 \geq 3 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

A) Write the corresponding LP for Phase I and Phase II.

**Exercise 04: (8)**

Soit le (PL) suivant :  $\text{Min } z = -5x_1 + 6x_2$

$$\begin{cases} -x_1 + 2x_2 \leq 5 \\ 5x_1 + 3x_2 = 6 \\ x_1 + 2x_2 \geq 3 \\ x_2 \geq 0 \\ x_1 \text{ unrestricted in sign} \end{cases}$$

- 1) A). Write the standard form of linear program to solve by simplex. B). Why do we need to add two artificial variables? C). What are the main variables and the slack variables?
- 2) A). Look for a realizable initial base solution. B). What are the base variables and non-base variables? C). Write the initial simplex table T0. D). Find the pivot.
- 3) A). Give the 2nd table.
- 4) A). Give the sequence of simplex tables and the optimal solution of this linear program.

**Exercise 01: (3)**

Which of the following constraints can be used in a Linear Program: (with justification)

- 1)  $e^{-x_1} + x_2 \leq 12$  ; 2)  $x_1 + |5x_2| x_3 = 0,5$  ; 3)  $|5x_1 + x_2| = + 0.666$  ; 4)  $-x_1 - x_2 \leq -5/3$  ;
- 5)  $\sum_{j=1}^N x_{ij} x_j \leq b_i$  ; 6)  $-x_3 - \ln(x_2) \leq -5/3$

**Exercise 02: (6)**

A) Determine all the basic solutions for the following equation system and specify for each solution

is feasible or not ?:

$$\begin{cases} x_1 + x_2 + x_3 + 2x_4 \geq 3 \\ -x_2 + x_3 + 2x_4 = 2 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

B) Write the Canonical form (Max) of this LP:

$$\text{MIN } Z = 3x_1 - 3x_2 + 7x_3 \begin{cases} x_1 + 9x_2 - 7x_3 \geq 5 \\ |5x_1 + x_2| = 3 \\ 1 \leq x_1 + x_2 - 7x_3 \leq 5 \\ x_1, x_2 \geq 0 \end{cases}$$

**Exercise 03 : (3)**

let consider the following LP:  $\text{MIN } Z = 3x_1 - 3x_2 + 7x_3 + 2x_4$

$$\begin{cases} x_1 + x_2 + x_3 - 3x_4 \geq 2 \\ -x_1 + x_3 + x_4 = 7 \\ x_1 + 4x_2 - 7x_3 \geq 3 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

A) Write the corresponding LP for Phase I and Phase II.

**Exercise 04: (8)**

Soit le (PL) suivant :  $\text{Min } z = -5x_1 + 6x_2$

$$\begin{cases} -x_1 + 2x_2 \leq 5 \\ 5x_1 + 3x_2 = 6 \\ x_1 + 2x_2 \geq 3 \\ x_2 \geq 0 \\ x_1 \text{ unrestricted in sign} \end{cases}$$

- 1) A). Write the standard form of linear program to solve by simplex. B). Why do we need to add two artificial variables? C). What are the main variables and the slack variables?
- 2) A). Look for a realizable initial base solution. B). What are the base variables and non-base variables? C). Write the initial simplex table T0. D). Find the pivot.
- 3) A). Give the 2nd table.
- 4) A). Give the sequence of simplex tables and the optimal solution of this linear program.

### Exercise 01: (3)

Which of the following constraints can be used in a Linear Program: (with justification)

- 1)  $e^{-x_1} + x_2 \leq 12$  ; 2)  $x_1 + |5x_2| x_3 = 0,5$  ; 3)  $|5x_1 + x_2| = + 0.666$  ; 4)  $-x_1 - x_2 \leq -5/3$  ;
- 5)  $\sum_{j=1}^N x_{ij} x_j \leq b_i$  ; 6)  $-x_3 - \ln(x_2) \leq -5/3$

### Exercise 02: (6)

A) Determine all the basic solutions for the following equation system and specify for each solution

is feasible or not ?:

$$\begin{cases} x_1 + x_2 + x_3 + 2x_4 \geq 3 \\ -x_2 + x_3 + 2x_4 = 2 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

B) Write the Canonical form (Max) of this LP:

$$\text{MIN } Z = 3x_1 - 3x_2 + 7x_3 \begin{cases} x_1 + 9x_2 - 7x_3 \geq 5 \\ |5x_1 + x_2| = 3 \\ 1 \leq x_1 + x_2 - 7x_3 \leq 5 \\ x_1, x_2 \geq 0 \end{cases}$$

### Exercise 03 : (3)

let consider the following LP:  $\text{MIN } Z = 3x_1 - 3x_2 + 7x_3 + 2x_4$

$$\begin{cases} x_1 + x_2 + x_3 - 3x_4 \geq 2 \\ -x_1 + x_3 + x_4 = 7 \\ x_1 + 4x_2 - 7x_3 \geq 3 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

A) Write the corresponding LP for Phase I and Phase II.

### Exercise 04: (8)

Soit le (PL) suivant :  $\text{Min } z = -5x_1 + 6x_2$

$$\begin{cases} -x_1 + 2x_2 \leq 5 \\ 5x_1 + 3x_2 = 6 \\ x_1 + 2x_2 \geq 3 \\ x_2 \geq 0 \\ x_1 \text{ unrestricted in sign} \end{cases}$$

- 1) A). Write the standard form of linear program to solve by simplex. B). Why do we need to add two artificial variables? C). What are the main variables and the slack variables?
- 2) A). Look for a realizable initial base solution. B). What are the base variables and non-base variables? C). Write the initial simplex table T0. D). Find the pivot.
- 3) A). Give the 2nd table.
- 4) A). Give the sequence of simplex tables and the optimal solution of this linear program.

1- Pour créer votre Startup, vous avez le choix entre se lancer en Solo ou s'associer avec d'autres personnes. **5pts**

- Citez les avantages et les inconvénients de chaque choix.

2- Définir brièvement avec un exemple de votre choix les coûts fixes, les coûts variables et les coûts mixtes. **4pts**

3- Pour son activité liée aux voitures particulières Mercedes s'adresse à des concessionnaires qui eux même vendent au client final. **3pts**

Quel est le type de marketing de :

- Mercedes
- Entre Mercedes et les concessionnaires
- Concessionnaires et clients

**Bon courage.**

**Exercise 01: (3)**

Which of the following constraints can be used in a Linear Program: (with justification)

- 1)  $e^{-x_1} + x_2 \leq 12$  ; 2)  $x_1 + |5x_2| x_3 = 0,5$  ; 3)  $|5x_1 + x_2| = + 0.666$  ; 4)  $-x_1 - x_2 \leq -5/3$  ;
- 5)  $\sum_{j=1}^N x_{ij} x_j \leq b_i$  ; 6)  $-x_3 - \ln(x_2) \leq -5/3$

**Exercise 02: (6)**

A) Determine all the basic solutions for the following equation system and specify for each solution

$$\text{is feasible or not? } \begin{cases} x_1 + x_2 + x_3 + 2x_4 \geq 3 \\ -x_2 + x_3 + 2x_4 = 2 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

B) Write the Canonical form (Max) of this LP:

$$\text{MIN } Z = 3x_1 - 3x_2 + 7x_3 \begin{cases} x_1 + 9x_2 - 7x_3 \geq 5 \\ |5x_1 + x_2| = 3 \\ 1 \leq x_1 + x_2 - 7x_3 \leq 5 \\ x_1, x_2 \geq 0 \end{cases}$$

**Exercise 03 : (3)**

let consider the following LP:  $\text{MIN } Z = 3x_1 - 3x_2 + 7x_3 + 2x_4 \begin{cases} x_1 + x_2 + x_3 - 3x_4 \geq 2 \\ -x_1 + x_3 + x_4 = 7 \\ x_1 + 4x_2 - 7x_3 \geq 3 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$

A) Write the corresponding LP for Phase I and Phase II.

**Exercise 04: (8)**

$$\text{Soit le (PL) suivant : Min } z = -5x_1 + 6x_2 \begin{cases} -x_1 + 2x_2 \leq 5 \\ 5x_1 + 3x_2 = 6 \\ x_1 + 2x_2 \geq 3 \\ x_2 \geq 0 \\ x_1 \text{ unrestricted in sign} \end{cases}$$

- 1) A). Write the standard form of linear program to solve by simplex. B). Why do we need to add two artificial variables? C). What are the main variables and the slack variables?
- 2) A). Look for a realizable initial base solution. B). What are the base variables and non-base variables? C). Write the initial simplex table T0. D). Find the pivot.
- 3) A). Give the 2nd table.
- 4) A). Give the sequence of simplex tables and the optimal solution of this linear program.

Questions : Pour chaque étude de cas essayez de répondre aux questions suivantes :

1. Illustrer avec une figure les frontières de chaque logiciel en tant que système ainsi que son environnement.
2. Quel type (SIA; SRI; SIAD; SABC; CAO; DAO; système technique ; système Embarqué...) assignez-vous au logiciel en question ?
3. Quels sont les parties prenantes (stakeholders) ?
4. Quelles sont les qualités attendues de chaque logiciel ?

Questions : Pour chaque étude de cas essayez de répondre aux questions suivantes :

1. Illustrer avec une figure les frontières de chaque logiciel en tant que système ainsi que son environnement.
2. Quel type (SIA; SRI; SIAD; SABC; CAO; DAO; système technique ; système Embarqué...) assignez-vous au logiciel en question ?
3. Quels sont les parties prenantes (stakeholders) ?
4. Quelles sont les qualités attendues de chaque logiciel ?

Questions : Pour chaque étude de cas essayez de répondre aux questions suivantes :

1. Illustrer avec une figure les frontières de chaque logiciel en tant que système ainsi que son environnement.
2. Quel type (SIA; SRI; SIAD; SABC; CAO; DAO; système technique ; système Embarqué...) assignez-vous au logiciel en question ?
3. Quels sont les parties prenantes (stakeholders) ?
4. Quelles sont les qualités attendues de chaque logiciel ?

## 5. Classe Inscription:

- ✓ Créez une classe **Inscription** avec les attributs privés **étudiant**, **cours** et **dateInscription**.
- ✓ Implémentez un constructeur **Inscription(étudiant: Étudiant, cours: Cours)**.
- ✓ Incluez des méthodes publiques pour obtenir l'étudiant (**getEtudiant(): Étudiant**) et la date d'inscription (**getDateInscription(): LocalDateTime**).

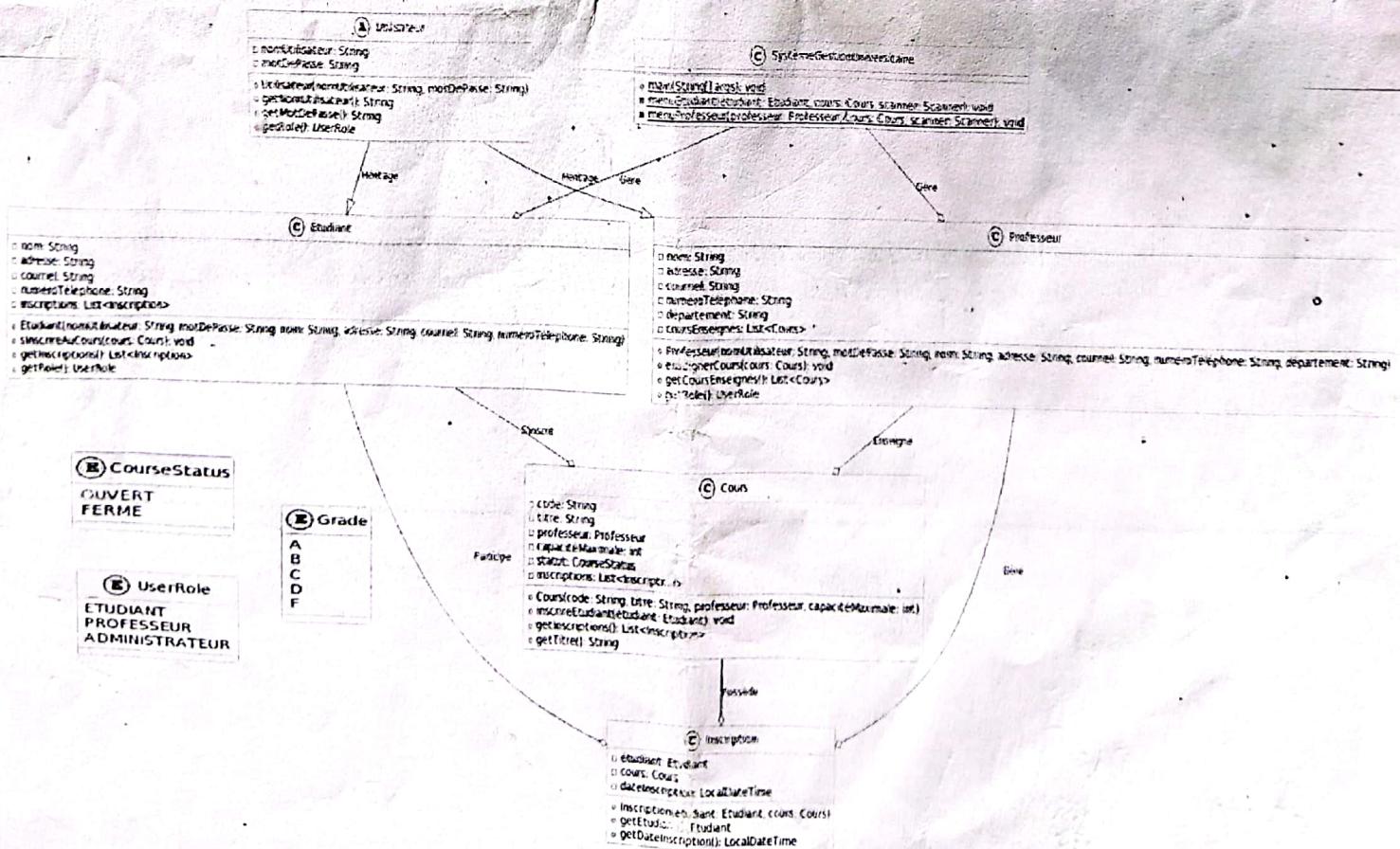
## 6. Classe SystèmeGestionUniversitaire (1 Pt) :

- ✓ Créez une classe **SystèmeGestionUniversitaire** avec des méthodes statiques pour l'application principale.
- ✓ Implémentez la méthode principale (**main(String[] args): void**) pour simuler le système de gestion universitaire.
- ✓ Incluez des méthodes pour gérer le menu étudiant (**menuEtudiant(étudiant: Étudiant, cours: Cours, scanner: Scanner): void**) et le menu professeur (**menuProfesseur(professeur: Professeur, cours: Cours, scanner: Scanner): void**).

## 7. Intégration du Menu:

- ✓ Intégrez le système de menu dans la classe **SystèmeGestionUniversitaire** pour gérer les interactions étudiant et professeur.

Le diagramme de classe :



## Interrogation de Génie Logiciel

**Exercice:** Rédigez un code Java basé sur le diagramme de classe fourni et les explications.

### 1. Classe Utilisateur:

- ✓ Créez une classe abstraite **Utilisateur** avec les attributs **nomUtilisateur** et **motDePasse**.
- ✓ Implémentez un constructeur abstrait **Utilisateur(nomUtilisateur: String, motDePasse: String)**.
- ✓ Incluez des méthodes publiques **getNomUtilisateur()** et **getMotDePasse()**.
- ✓ Incluez une méthode abstraite **getRole(): UserRole**.

### 2. Classe Étudiant:

- ✓ Héritez la classe **Utilisateur** pour créer la classe concrète **Étudiant**.
- ✓ Ajoutez les attributs privés **nom**, **adresse**, **courriel** et **numéroTéléphone**.
- ✓ Implémentez un constructeur **Étudiant(nomUtilisateur: String, motDePasse: String, nom: String, adresse: String, courriel: String, numéroTéléphone: String)**.
- ✓ Fournissez des méthodes publiques getter et setter pour les attributs supplémentaires.

### 3. Classe Professeur:

- ✓ Héritez la classe **Utilisateur** pour créer la classe concrète **Professeur**.
- ✓ Ajoutez les attributs privés **nom**, **adresse**, **courriel**, **numéroTéléphone** et **département**.
- ✓ Implémentez un constructeur **Professeur(nomUtilisateur: String, motDePasse: String, nom: String, adresse: String, courriel: String, numéroTéléphone: String, département: String)**.
- ✓ Fournissez des méthodes publiques getter et setter pour les attributs supplémentaires.

### 4. Classe Cours:

- ✓ Créez une classe **Cours** avec les attributs privés **code**, **titre**, **professeur**, **capacitéMaximale**, **statut** et **inscriptions**.
- ✓ Implémentez un constructeur **Cours(code: String, titre: String, professeur: Professeur, capacitéMaximale: int)**.
- ✓ Incluez des méthodes publiques pour inscrire un étudiant (**inscrireÉtudiant(étudiant: Étudiant): void**), obtenir les inscriptions (**getInscriptions(): List<Inscription>**) et obtenir le titre (**getTitre(): String**).

**Software Engineering Exam Paper**

Duration 1h30

Unauthorized documents

**Exercise 1: (8 points)**

Provide short and summarized answers to the following questions:

1. Define the term "model" in the context of software engineering. How does it help in understanding complex systems? (1.5p)
2. Explain the purpose of a collaboration diagram? (What is it used for) 1.5p
3. What are the key components of software? (2.5p)
4. What are the main features that distinguish software development from the production of other products? (2.5p)

**Exercise 2: (6 points)**

We are planning to create a special software for managing computer projects in our company. The system allows all employees to check information about the projects. Whether they want to see if a specific project exists or explore information, they need to log in with basic details like their name and department for future statistics.

Engineers, who are responsible for projects, have extra privileges. They can do different tasks like adding, removing, or changing project details. However, these sensitive tasks need a deeper login. Engineers must provide their name, department, and a password, which is checked with the personnel management system.

Every action, whether it's checking or updating, is recorded in an access log. Also, users can choose to print the documents they accessed.

1. Visualiser le flux des opérations dans le système à travers le diagramme des cas d'utilisation. (6p)

**Exercise 3: (6 points)**

A retail store wants to streamline its inventory management system. Here are the key functionalities:

- For each product, the system needs to store the product name, category, and price.
- Each product can have multiple units in stock, each identified by a unique serial number.
- The store also maintains information about suppliers, including their names and contact details.
- Each product unit in stock is associated with a specific supplier.
- The system records sales transactions, including the date of sale and the quantity sold for each product.
- There is a need to track customer information, including their names and contact details.
- Every sales transaction is linked to a specific customer.

1. Design a class diagram to illustrate the structure of this inventory management system. (6p)

Best wishes

Sujet d'examen Génie Logiciel

Durée 1h30

Documents non-autorisés

**Exercice 1: (8 points)**

Fournissez des réponses courtes et résumées aux questions suivantes :

1. Définissez le terme "modèle" dans le contexte du génie logiciel. Comment cela aide-t-il à comprendre les systèmes complexes ? (1.5p)
2. Expliquez la finalité d'un diagramme de collaboration. (À quoi sert-il ?) 1.5p
3. Quels sont les composants clés d'un logiciel ? (2.5p)
4. Quelles sont les principales caractéristiques qui distinguent le développement logiciel de la production d'autres produits ? (2.5p)

**Exercice 2: (6 points)**

Nous prévoyons de créer un logiciel spécial pour la gestion des projets informatiques dans notre entreprise. Le système permet à tous les employés de vérifier des informations sur les projets. Que ce soit pour voir si un projet spécifique existe ou pour explorer des informations, ils doivent se connecter avec des détails de base tels que leur nom et leur département pour les statistiques futures.

Les ingénieurs, responsables des projets, ont des priviléges supplémentaires. Ils peuvent effectuer différentes tâches telles que l'ajout, la suppression ou la modification des détails du projet. Cependant, ces tâches sensibles nécessitent une connexion plus approfondie. Les ingénieurs doivent fournir leur nom, leur département et un mot de passe, qui est vérifié avec le système de gestion du personnel.

Chaque action, qu'il s'agisse de vérifier ou de mettre à jour, est enregistrée dans un journal d'accès. De plus, les utilisateurs peuvent choisir d'imprimer les documents auxquels ils ont accédé.

1. Visualiser le flux des opérations dans le système à travers le diagramme des cas d'utilisation. (6p)

**Exercice 3: (6 points)**

Un magasin de détail souhaite rationaliser son système de gestion des stocks. Voici les fonctionnalités clés :

- Pour chaque produit, le système doit stocker le nom du produit, la catégorie et le prix.
  - Chaque produit peut avoir plusieurs unités en stock, chacune identifiée par un numéro de série unique.
  - Le magasin conserve également des informations sur les fournisseurs, y compris leurs noms et coordonnées.
  - Chaque unité de produit en stock est associée à un fournisseur spécifique.
  - Le système enregistre les transactions de vente, y compris la date de vente et la quantité vendue pour chaque produit.
  - Il est nécessaire de suivre les informations sur les clients, y compris leurs noms et coordonnées.
  - Chaque transaction de vente est liée à un client spécifique.
1. Concevoir un diagramme de classe pour illustrer la structure de ce système de gestion des stocks. (6p)



## Examen de Compilation

### Exercice 1. (4 pts)

- Donner une description en français (ou en arabe) des langages dénotés par les expressions régulières suivantes définies sur l'alphabet  $\Sigma = \{a, b\}$  :
  - $a^+ | b^+$
  - $a^+b^+$
- Donner une expression régulière pour les langages suivants sur l'alphabet  $\{a, b\}$ .
  - L'ensemble des mots de longueur impaire*
  - L'ensemble des mots où tous les blocs de a sont de longueur paire*

### Exercice 2. (4 pts)

Donner les grammaires générant les langages suivants :

- $L_2 = \{a^n b^{2n} \mid n \geq 0\}$
- $L_1 = \{a^n b^m \mid n \neq m, n, m \geq 0\}$

### Exercice 3. (6 pts)

On considère la grammaire  $G$  suivante :

$$\begin{aligned} G: \quad S &\rightarrow \{R\} \mid a \\ R &\rightarrow K \mid \epsilon \\ K &\rightarrow K, S \mid S \end{aligned}$$

- Éliminer la récursivité gauche dans la grammaire.
- Créer la table d'analyse LL(1) pour la grammaire produite en question 1.
- Donner la trace et le résultat de l'analyse de la phrase :  $\{\{a, a\}, a\}$

### Exercice 3. (6 pts)

On considère la grammaire  $G$  suivante :

$$\begin{aligned} G: \quad S &\rightarrow \{R\} \mid a \\ R &\rightarrow K \mid \epsilon \\ K &\rightarrow K, S \mid S \end{aligned}$$

- Construire l'automate LR(0) de la grammaire  $G$ .
- Construire la table d'analyse SLR(1) de cette grammaire
- Donner la trace et le résultat de l'analyse de la phrase :  $\{\{a, a\}, a\}$

Figure 1

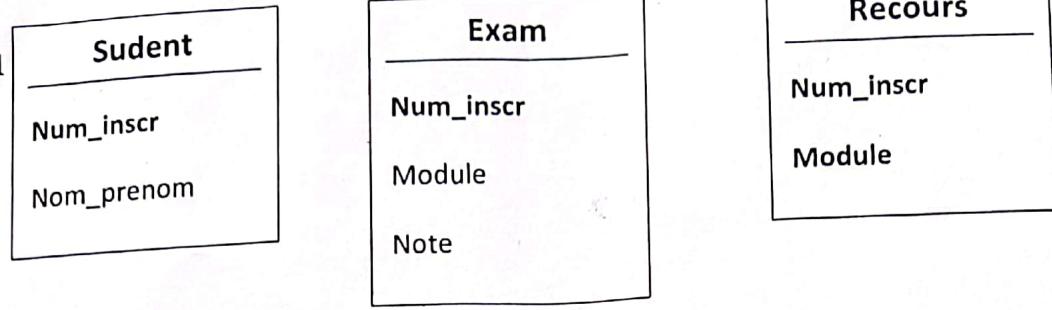


Figure 2

Num Inscr num\_nscrEdit

Password pwdEdit

**login\_Btn**

This image shows a user interface for logging in. It features two input fields: one for the number of inscription labeled "Num Inscr" and another for the password labeled "Password". Below these fields is a rounded rectangular button labeled "login\_Btn".

Figure 3

44.00

num\_inscr num\_Edit

Nom\_Prenom nom\_prenomEdit

**Update\_Btn**

Système D'exploitation 12

Dev Apps Mobiles 12

Sécurité Informatique 12

Génie logiciel 12

Donnés semi-structurées 12

**moduleTv**

**recoursBtn**

**noteTv**

A screenshot of a software application. At the top left, it displays "44.00". Below this, there are two text input fields: "num\_inscr" and "Nom\_Prenom", each with an associated edit field ("num\_Edit" and "nom\_prenomEdit"). A "Update\_Btn" button is positioned below these fields. The main area of the screen lists five modules with their respective credit values: "Système D'exploitation" (12), "Dev Apps Mobiles" (12), "Sécurité Informatique" (12), "Génie logiciel" (12), and "Donnés semi-structurées" (12). To the right of the "Sécurité Informatique" entry, there is a bracket labeled "Info Etudiant" grouping the "recoursBtn" and "noteTv" buttons. A bracket on the far left groups the "moduleTv" button with the "Système D'exploitation" entry.

**Examen : Développement des Applications Mobiles**  
**(Documents cours & TPs autorisés : 1h 30)**

**Exercice 1**

On veut réaliser une application mobile permettant aux étudiants de consulter leurs notes. Pour simplifier, on suppose qu'il existe une base SQLite locale « **studentdb** » représentant une copie d'une vue d'une base distante (database replication). La figure 1 présente un schéma simplifié de la base (on ne vous demande pas de créer cette base).

A la première utilisation, l'étudiant doit saisir son numéro d'inscription et son mot de passe via l'activité « **login\_Activity** » de la figure 2. Ces informations seront stockées dans un shared preference pour une utilisation ultérieure.

Une fois le bouton « login » est cliqué, l'activity « **Mai\_activity** » est lancée (Figure 3). Au démarrage de cette activité, on lit le contenu du shared preference **credentials\_pref** pour extraire le numéro d'inscription. Ce dernier permet de formuler une requête pour lire le nom\_prénom et les notes de l'étudiant associé au numéro d'inscription préalablement saisi à partir de la base **studentdb**. Les notes sont affichées dans une listView en utilisant un adapter « **noteAdapter** ».

- Le bouton **Update.Btn** de la figure 3 permet à l'étudiant de corriger son nom et prénom et mettre à jour la base.
- Le bouton **RecourBtn**  de la figure 3 permet de formuler un recours contenant le numéro d'inscription de l'étudiant et le module concerné. Ce recours sera inséré dans la table Recours de la base.

**Questions :**

- 1- Ecrire le code XML associé à la partie Info Etudiant de la figure 3.
- 2- Ecrire la méthode **ReadPreferences()** permettant de lire le **num\_inscr** de l'utilisateur à partir du shared preference nommé « **connexion\_pref** ».
- 3- Ecrire le code de la méthode **getStudentName(String num\_inscr)** permettant de lire le **nom\_prenom** de l'étudiant associé au **num\_inscr**.
- 4- Ecrire la classe « **Exam** » du datamodel associé à la table **Exam**.
- 5- Ecrire le code de la fonction **ArrayList<Exam> getStudentMarks(String num\_inscr)** permettant de lire les notes de l'étudiant associé au **num\_inscr**.
- 6- Ecrire le code de l'adapter **noteAdapter**.
- 7- Ecrire le code de la méthode **InsertRecour** permettant d'insérer un recours dans la table **Recours**.

**Question supplémentaire**

- Ecrire le code du callback **OnCreate** de l'activité **Main\_activity**

Bon courage



## Examen Semestriel

Documents non autorisés – durée (1h :30)

### Exercice 3 (7points)

On veut écrire d'une manière infinie le mot « GAZA ». Pour ce faire on dispose des trois process suivants :

Pg()	Pa()	Pz()
while(true)    write('G') ;	while(true)    write('A') ;	while(true)    write('Z') ;

- On vous demande de synchroniser les trois process en utilisant les moniteurs.
- Le code précédent reste-t-il valide si l'on a deux processus de type A ? Justifier votre réponse.

### Exercice 3 (3 points) : Interblocage

Soit un système se composant de quatre process et trois ressources. A un instant t un process a fait une requête dont l'état simulé du système est le suivant :

Available		Allocate			Announce		
		R1	R2	R3	R1	R2	R3
1,1,0		0	1	0	2	2	0
	P1	1	0	0	1	0	1
	P2	1	0	1	1	2	1
	P3	0	1	0	1	1	0
	P4						

- 1- Etablir la matrice NEED?
- 2- Cet état est-il flable ? Justifier votre réponse ?

Bon courage



Exercice 1 (3 points) : Une solution au problème du pont à voie unique par régions critique a été proposée comme suit :

```
const int N = 10;
struct PONT {
    int nbAtt1, nbAtt2, nbSurpont;
    {'R', 'V'} feux1, feux2;
} pont;
pont.nbAtt1 = pont.nbAtt2 = pont.nbSurpont = 0; pont.feux1 = pont.feux2 = 'V';
```

### Dt1()

```
Region pont do {
    nbAtt1++;
    await( feux2 == 'V' && nbSurpont <= N)
    nbAtt1--;
    if(nbSurpont == 0) feux2 = 'R';
    nbSurpont++;
}
```

### Ft1()

```
Region pont do {
    nbSurpont--;
    if(nbSurpont == 0){
        feux2 = 'V'
    }
    else{
        feux2 = 'R';
    }
}
```

Question : répondre par VRAI/FAUX à chaque proposition, une fausse réponse entraîne une note négative.

- 1- La capacité du pont n'est jamais atteinte (nbSurpont n'égalera jamais N)
- 2- La capacité du pont peut être dépassée (nbSurpont peut être > N)
- 3- Il y a une possibilité de collision (au moins deux véhicules de deux sens circulent en même temps)
- 4- Il y a une possibilité qu'une classe (de sens1 ou sens2) n'accède jamais au pont
- 5- La solution impose une priorité FIFO entre les véhicules des deux classes.
- 6- Toutes les propositions précédentes sont fausses.

Exercice 2 (8 points) : pour fabriquer un objet X composé de 2 pièces A et B. On dispose de trois robots R1, R2 et R3:

- R1 : fabrique une pièce A et la dépose dans le bac (ou box) B1.
- R2 : fabrique une pièce B et la dépose dans le bac (ou box) B2.
- R3 : retire une pièce A du bac B1 et une pièce B du bac B2 et fait l'assemblage des deux pour produire un objet X et le dépose dans un bac B3.

```
R1()
while(true) {
    A = fabriquerA();
    Deposer(A, B1);
}
```

```
R2()
while(true) {
    B = fabriquerB();
    Deposer(B, B2);
}
```

```
R3()
while(true) {
    A = Retirer(B1);
    B = Retirer(B2);
    X = fabriquerX(A, B)
    Deposer(X, B3)
}
```

- 1- Synchroniser les trois robots en utilisant les sémaphores.

Pour accélérer l'opération du montage, on ajoute un quatrième robot R4 pour aider R3.

- 2- Faire les modifications nécessaires au code précédent pour prendre en compte cette situation



## 5. Enrollment Class:

- ✓ Create a class **Enrollment** with private attributes **student**, **course**, and **enrollmentDate**.
- ✓ Implement a constructor **Enrollment(student: Student, course: Course)**.
- ✓ Include public methods for getting the student (**getStudent(): Student**) and enrollment date (**getEnrollmentDate(): LocalDateTime**).

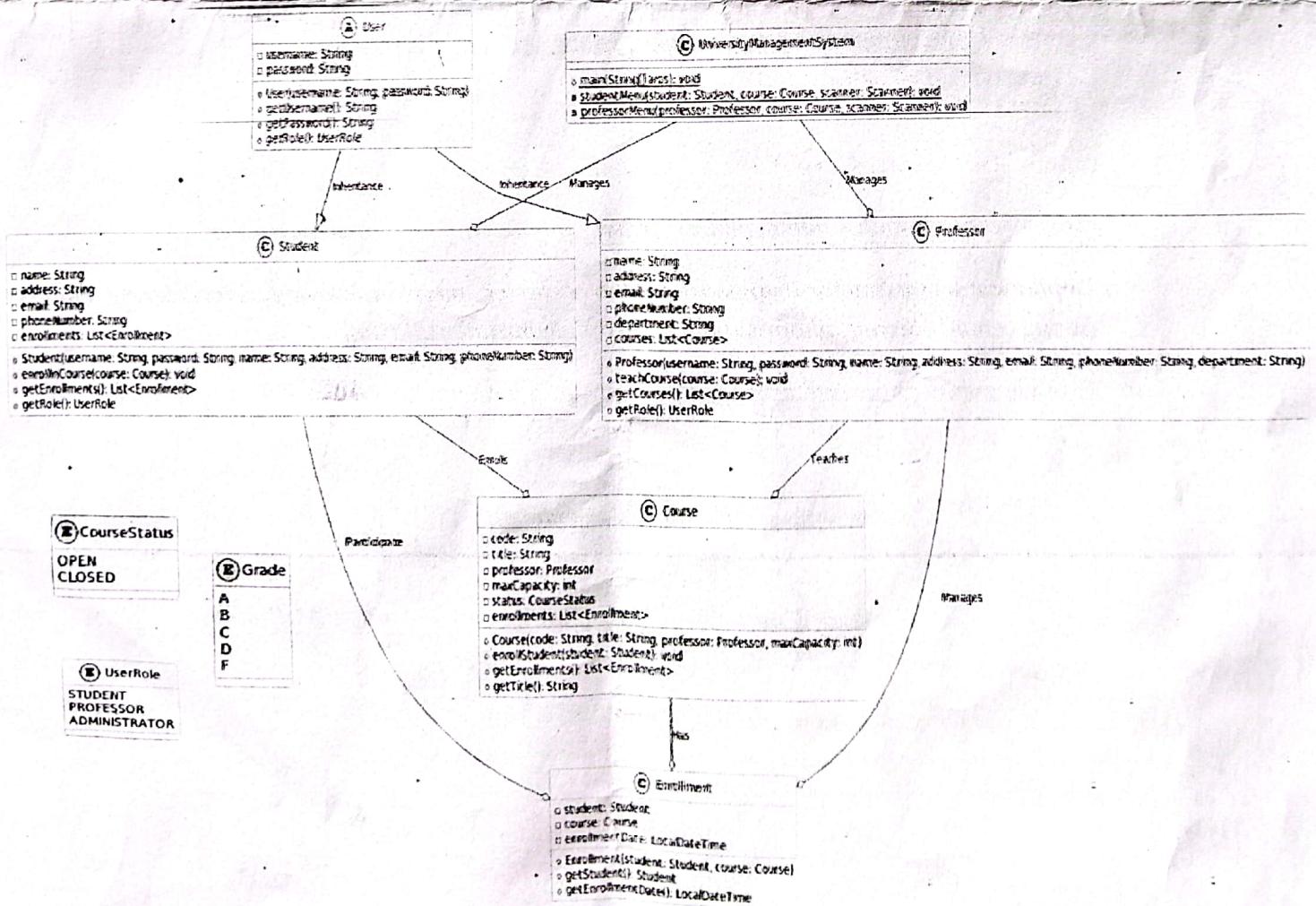
## 6. UniversityManagementSystem Class:

- ✓ Create a class **UniversityManagementSystem** with static methods for the main application.
- ✓ Implement the main method (**main(String[] args): void**) to simulate the university management system.
- ✓ Include methods for handling the student menu (**studentMenu(student: Student, course: Course, scanner: Scanner): void**) and professor menu (**professorMenu(professor: Professor, course: Course, scanner: Scanner): void**).

## 7. Menu Integration:

- ✓ Integrate the menu system into the **UniversityManagementSystem** to handle student and professor interactions.

The class diagram:



## Software Engineering Interrogation

**Exercise:** Write a Java code based on the provided class diagram and explanations

### 1. User Class:

- ✓ Create an abstract class *User* with attributes *username* and *password*.
- ✓ Implement an abstract constructor *User(username: String, password: String)*.
- ✓ Include public methods *getUsername()* and *getPassword()*.
- ✓ Include an abstract method *getRole(): UserRole*.

### 2. Student Class:

- ✓ Extend the *User* class to create a concrete class *Student*.
- ✓ Add private attributes *name*, *address*, *email*, and *phoneNumber*.
- ✓ Implement a constructor *Student(username: String, password: String, name: String, address: String, email: String, phoneNumber: String)*.
- ✓ Provide public getter and setter methods for the additional attributes.

### 3. Professor Class:

- ✓ Extend the *User* class to create a concrete class *Professor*.
- ✓ Add private attributes *name*, *address*, *email*, *phoneNumber*, and *department*.
- ✓ Implement a constructor *Professor(username: String, password: String, name: String, address: String, email: String, phoneNumber: String, department: String)*.
- ✓ Provide public getter and setter methods for the additional attributes.

### 4. Course Class:

- ✓ Create a class *Course* with private attributes *code*, *title*, *professor*, *maxCapacity*, *status*, and *enrollments*.
- ✓ Implement a constructor *Course(code: String, title: String, professor: Professor, maxCapacity: int)*.
- ✓ Include public methods for enrolling a student (*enrollStudent(student: Student): void*), getting enrollments (*getEnrollments(): List<Enrollment>*), and getting the title (*getTitle(): String*).

```

    public ItemPhoto photo;
    public ItemPhoto(Caption caption, String description, String telNum, Drawable photo) {
        this.caption = caption;
        this.description = description;
        this.telNum = telNum;
        this.photo = photo;
    }
}

```

- 2- Ecrire le code associé au bouton « Capture Photo » de la figure 2.

```

ImageView itemPhoto = findViewById(R.id.itemPhoto);
Button capturBtn = findViewById(R.id.capturePhoto);
capturBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Drawable photo = capturePhoto();
        itemPhoto.setImageDrawable(photo);
    }
});

```

- 3- Ecrire le code associé à « send Declaration » de la figure 2.

```

Button sendBtn = findViewById(R.id.sendDeclarationBtn);
sendBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        EditText captionEdit = findViewById(R.id.captionEdit);
        EditText descrEdit = findViewById(R.id.descrEdit);
        EditText telEdit = findViewById(R.id.telEdit);
        Drawable itemPhoto = findViewById(R.id.itemPhoto);

        Item item = new Item(captionEdit.getText().toString(),
                             descrEdit.getText().toString(),
                             telEdit.getText().toString(),
                             itemPhoto.getCurrent());
        Insert(item);
    }
});

```

- 4- Ecrire le code associé à un clic sur une élément de la liste de la figure 3.

```

ListView itemsList = findViewById(R.id.itemsList);
itemsList.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        Item item = lostItemsList.get(i);
        Intent intent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:" + item.telNum));
        startActivity(intent);
    }
});

```

- 5- Ecrire l'interface de code associé à la classe du composant DAO (SQL Data Access Object) dans lequel vous définissez les deux méthodes :

- `insert(Item item)`
- `List <Item> getAllDeclarations()`

```

public class Item {
    public String caption;
    public String description;
    public String telNum;
    public Drawable photo;
}

}

```

- 2- Ecrire le code associé au bouton « **Capture Photo** » de la figure 2.

```

ImageView itemPhoto = findViewById(R.id.itemPhoto);
Button captureBtn = findViewById(R.id.capturePhoto);
captureBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Drawable photo = capturePhoto();
        itemPhoto.setImageDrawable(photo);
    }
});

```

- 3- Ecrire le code associé à « **send Declaration** » de la figure 2.

```

Button sendBtn = findViewById(R.id.sendDeclarationBtn);
sendBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        EditText captionEdit = findViewById(R.id.captionEdit);
        EditText descrEdit = findViewById(R.id.descrEdit);
        EditText telEdit = findViewById(R.id.telEdit);
        Drawable itemPhoto = findViewById(R.id.itemPhoto);

        Item item = new Item(captionEdit.getText().toString(),
                             descrEdit.getText().toString(),
                             telEdit.getText().toString(),
                             itemPhoto.getCurrent());
        Insert(item);
    }
});

```

- 4- Ecrire le code associé à un clic sur une élément de la liste de la figure 3.

```

ListView itemsList = findViewById(R.id.itemsList);
itemsList.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        Item item = lostItemsList.get(i);
        Intent intent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:" + item.telNum));
        startActivity(intent);
    }
});

```

- 5- Ecrire l'interface de code associé à la classe du composant **DAO** (**SQL Data Access Object**) dans lequel vous définissez les deux méthodes :

- insert(Item item)**
- List <Item> getAllDeclarations()**

Examen : Développement des Applications Mobiles  
Documents cours & TP autorisés : 1h 30

Exercice 1 (4 points) : On veut créer une interface Android dans laquelle les trois centres des widgets Btn1 et Btn2 et le bouton « FIX\_ME\_1 » sont alignés suivant une droite (figure 1) quel que soit les positions de Btn1 et Btn2.

Ecrire le code XML, permettant de générer le **bouton FIX\_ME\_1** en respectant cette contrainte.

```
<Button  
    android:id="@+id/Fix_Me_1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Fix_Me_1"  
    app:layout_constraintBottom_toTopOf="@+id/BTN2"  
    app:layout_constraintEnd_toEndOf="@+id/BTN1"  
    app:layout_constraintStart_toStartOf="@+id/BTN2"  
    app:layout_constraintTop_toBottomOf="@+id/BTN1" />
```

Exercice 3 :

On veut réaliser une application mobile permettant d'aider les gens à trouver leurs objets perdus. On s'intéresse ici à la partie « déclaration des objets perdus » consistant en deux principales activités :

- **Lost\_Activity** : permettant à un user de signaler un objet perdu (**Fig. 2**).

- **LostItems\_Activity** : permettant à un user d'explorer la liste de tous les objets déclarés perdus (**Fig. 3**).

⇒ Pour simplifier, un objet « item » est identifié par une désignation ou caption (téléphone, montre, somme d'argent, ...), une description, une photo et un numéro de téléphone de la personne qui l'a déclaré.

⇒ Un clic sur le bouton « Capture Photo » permet de capturer une photo de l'objet et l'associer à l'ImageView de la figure 2.

⇒ Un clic sur le bouton « send Declaration » permet de créer une nouvelle déclaration et l'insérer dans la base de données.

⇒ Un clic sur un élément de la liste de la figure 3 permet de faire un appel téléphonique à la personne qui a signalé la perte de l'objet sélectionné.

NB : Vous disposez des méthodes suivantes :

- **Drawable capturePhoto()** : permet de capturer une photo en utilisant la caméra par défaut et renvoie une image de type Drawable.
- **insert(Item item)** : permet d'insérer un objet perdu 'item' dans la base.
- **List <Item> getAllDeclarations()** : permet de retourner tous les objets déclarés perdus.

Questions :

- 1- Ecrire la classe « **Item** » sous forme d'une « **Entity** »

```
@Entity  
public class Item {  
    public String caption;  
    public String description;  
    public String telNum;
```

## CORRECTION

- 1) Le code XML base LINEARLAYOUT (0.25 pour chaque linearlayout complet, 0.5 pour chaque widget complet)

```
LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Nom" />

    <EditText
        android:id="@+id/nomEdit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="3"
        android:hint="votre nom ici"
        android:inputType="textPersonName" />

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Prénom" />

    <EditText
        android:id="@+id/prenomEdit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="3"
        android:hint="votre prénom ici"
        android:inputType="textPersonName" />

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

Exercice 1 : on veut créer une application mobile permettant aux étudiants de consulter leurs notes à distance.

Cette application consiste en une activité, nommée **MainActivity**, permettant l'introduction du nom, et le prénom et la sélection de l'année d'étude à partir d'une liste déroulante (Figure 1). Les notes sont affichées dans une liste.

Un clic sur le bouton « AFFICHER » de la figure 1 permet de créer une liste, nommée **notesList** de type **ArrayList** affichant chaque module avec sa note, contenant toutes les notes des modules de l'étudiant. L'affichage de cette liste est rendu par une **ListView**, ayant l'id **notesListView**.



Figure 1

#### Questions (5+2+4+5+4 pts):

1. Ecrire le code XML permettant de générer l'interface de la figure 1 en utilisant le **LinearLayout**. (Contentez-vous des attributs de width, height et weight).
2. Ecrire la classe **Module**.
3. Ecrire le code XML définissant les entrées de la liste déroulante (les éléments sont : **Première Année**, **Deuxième Année**, **Troisième Année**)
4. Ecrire la class **ModuleAdapter** permettant l'affichage de la **ListView**.
5. Ecrire le callBack **onCreate** de l'activité dans lequel vous instanciez : **notesList**, **notesListView** et **adapter** (de type **ModuleAdapter**).

Bon courage

1- Code XML de la figure 1 (5 POINTS)

<ConstraintLayout>

```
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <Button
        android:id="@+id/searchBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Chercher"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/prodEdit"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/prodEdit"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="produits à chercher"
        app:layout_constraintEnd_toStartOf="@+id/button"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</ConstraintLayout>
```

2- Code de la classe Product (2 POINTS)

```
package com.mad.examen2021;

public class Product {
    String prodName;
    double price;

    public Product(String prodName, double price) {
        this.prodName = prodName;
        this.price = price;
    }

    public String getProdName() {
        return prodName;
    }

    public void setProdName(String prodName) {
        this.prodName = prodName;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }
}
```

Exercice 1 On s'intéresse à la réalisation d'une application mobile pour un magasin permettant la commande d'un produit à distance. Cette application consiste en une seule activité, nommée **MainActivity**, permettant la recherche (Figure 1) et la commande d'un produit (Figure 2).

Un clic sur le bouton « **CHERCHER** » de la figure 1 permet de créer une liste, nommée **productList** de type **ArrayList**, contenant tous les produits répondant au critère de recherche tapé dans l'**EditText** à gauche du bouton. L'affichage de cette liste est rendu par une **ListView**, ayant l'id **prodListView**, dont l'interface d'un seul élément est donnée par la figure 2.

Un clic sur le bouton « **COMMANDER** » permet de commander un produit de la liste en envoyant un message court au service commercial du magasin contenant le produit à commander et quelques informations de l'utilisateur (numéro de téléphone, ses coordonnées (GPS ou autres)).

- Le numéro de téléphone du magasin est un paramètre global nommé **telMagasin** de type **String**.
- Une fonction « **String getUserInfo()** » permet de récupérer les informations globales de l'utilisateur.

#### Questions (5+2+4+5+4 pts):

- Ecrire le code XML permettant de générer l'interface de la figure 1 en utilisant le **ConstraintLayout**. (Contentez-vous des attributs de width, height et les contraintes).
- Ecrire la classe **Product** représentant le data Model de l'adaptateur de la liste.
- Ecrire la méthode **commander(String ProductName)** permettant de commander un produit.
- Ecrire la class **productAdapter** permettant l'affichage de la listView.
- Ecrire le **callback onCreate** de l'activité **MainActivity** qui instancie **productList**, **prodListView** et **adapter** (le type **productAdapter**).

#### Question supplémentaire (2pts):

- Ecrire le code XML associé à l'interface de la figure 2 en utilisant le **LinearLayout**.

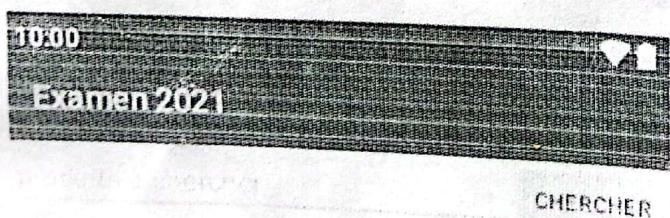


Figure 1

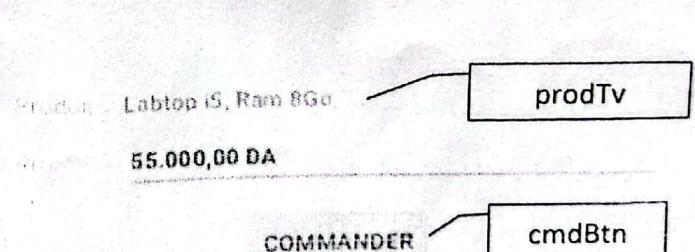


Figure 2

Bon courage

```
    return v;
```

### 5- Le callback oncreate (5 POINTS)

```
ListView prodListView;
ArrayList<Product> productList;
ProductAdapter prodAdapter;
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}

productList = new ArrayList<>();
prodListView = findViewById(R.id.prodListView);
prodAdapter = new ProductAdapter(this, productList);
prodListView.setAdapter(prodAdapter);
```

3- Code de la méthode commander (4 POINTS)

```
private void commander(String prodName) {  
    String info = getuserInfo() + " - " + prodName;  
    SmsManager smsManager = SmsManager.getDefault();  
    smsManager.sendTextMessage(telMagasin, null, info, null, null);
```

4- Code du productAdapter (5 POINTS)

```
public class productAdapter extends BaseAdapter {  
  
    private final Context context;  
    private final ArrayList<Product> list;  
    private String telMagasin;  
  
    public productAdapter(Context context, ArrayList<Product> list) {  
        this.context = context;  
        this.list = list;  
    }  
  
    public int getCount() {  
        return list.size();  
    }  
  
    public Product getItem(int position) {  
        return list.get(position);  
    }  
  
    public long getItemId(int position) {  
        return position;  
    }  
  
    @Override  
    public View getView(int position, View convertView, ViewGroup parent) {  
        LayoutInflater inf = LayoutInflater.from(context);  
        View v = inf.inflate(R.layout.product_item, null);  
  
        TextView prodnameTv = v.findViewById(R.id.prodTv);  
        TextView priceTV = v.findViewById(R.id.priceTv);  
        Button cmdBtn = v.findViewById(R.id.cmdBtn);  
        prodnameTv.setText(getItem(position).getProdName());  
        priceTV.setText(getItem(position).getPrice() + "");  
  
        final String prodName = prodnameTv.getText().toString();  
        cmdBtn.setOnClickListener(new View.OnClickListener() {  
  
            public void onClick(View v) {  
                commander(prodName);  
            }  
        });  
    }
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    listItemsList = new ArrayList();
    listView = findViewById(R.id.listView);
    adapter = new ListItemsAdapter(listItemsList, this);
    listView.setAdapter(adapter);
}
```

8. L'application étant simplifiée, que proposer vous pour améliorer ses fonctionnalités (pousser la conception) ?

Bon courage

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
}
```

```
    lostItemList = new ArrayList<String>();  
    listView = findViewById(R.id.itemList);  
    adapter = new LostItemAdapter(lostItemList, this);  
    listView.setAdapter(adapter);  
}
```

8. L'application étant simplifiée, que proposer vous pour améliorer ses fonctionnalités (pousser la conception)?

Bon courage

protected void onCreate(Bundle savedInstanceState) {  
super.onCreate(savedInstanceState);  
setContentView(R.layout.activity\_main);

```
listItemList = new ArrayList<String>();  
listView = findViewById(R.id.listView);  
adapter = new ListItemsAdapter(listItemList, this);  
listView.setAdapter(adapter);
```

- 8- L'application étant simplifiée, que proposer vous pour améliorer ses fonctionnalités (pousser la conception)?

Bon courage

~~Protected methods from your superclass that you want to override~~  
super.onCreate(savedInstanceState);  
~~Call super.onCreate(savedInstanceState) at the end of your onCreate method.~~

```
listItemList = new ArrayList();
listView = findViewById(R.id.listView);
adapter = new ListItemsAdapter(listItemList, this);
listView.setAdapter(adapter);
```

8. L'application étant simplifiée, que proposer vous pour améliorer ses fonctionnalités (pousser la conception)?

Bon courage

protect void onCreate(Bundle savedInstanceState) {  
super.onCreate(savedInstanceState);  
setContentView(R.layout.activity\_main);  
List<String> listItems = new ArrayList();

```
listItemList = new ArrayAdapter()  
listView = findViewById(R.id.itemList);  
adapter = new ListViewAdapter(listItemList, this);  
listView.setAdapter(adapter);  
}
```

8. L'application étant simplifiée, que proposer vous pour améliorer ses fonctionnalités (pousser la conception) ?

Bon courage

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);  
    setSupportActionBar(toolbar);  
}
```

```
lostItemList = new ArrayList<String>();  
listView = findViewById(R.id.itemsList);  
adapter = new LostItemAdapter(lostItemList, this);  
listView.setAdapter(adapter);
```

8. L'application étant simplifiée, que proposer vous pour améliorer ses fonctionnalités (pousser la conception) ?

Bon courage

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

```
lostItemsList = new ArrayList();
listView = findViewById(R.id.itemList);
adapter = new LostItemsAdapter(lostItemsList, this);
listView.setAdapter(adapter);
```

8- L'application étant simplifiée, que proposer vous pour améliorer ses fonctionnalités (pousser la conception) ?

Bon courage

```

class
public interface LostData {
    void insert
    void update(item);
}

Query("SELECT * from Item")
listItem.getAllDeclarations();

```

- 6- Ecrire la classe **lostItemsAdapter** permettant d'afficher le contenu d'une liste d'Items (**List <Item>**). Il faut respecter les Ids déclarés dans la figure 3.

```

public class lostItemsAdapter extends BaseAdapter {

    List<Item> lostItemsList;
    Context context;

    public lostItemsAdapter(List<Item> lostItemsList, Context context) {
        this.lostItemsList = lostItemsList;
        this.context = context;
    }

    @Override
    public int getCount() {
        return lostItemsList.size();
    }

    @Override
    public Item getItem(int i) {
        return lostItemsList.get(i);
    }

    @Override
    public long getItemId(int i) {
        return i;
    }

    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        LayoutInflator inflater = LayoutInflator.from(context);
        View v = inflater.inflate(R.layout.item_lost, null);

        Item item = getItem(i);

        TextView captionV = v.findViewById(R.id.captionTextview);
        captionV.setText(item.caption);

        TextView descrTv = v.findViewById(R.id.descrTextview);
        descrTv.setText(item.description);

        ImageView phView = v.findViewById(R.id.imgageView);
        phView.setImageDrawable(item.photo);

        return v;
    }
}

```

- 7- Ecrire le code associé au callback **onCreate** de l'activité **LostItems \_Activity** (Fig. 3).

```

List<Item> lostItemsList;
private View listView;
private lostItemsAdapter adapter;

```

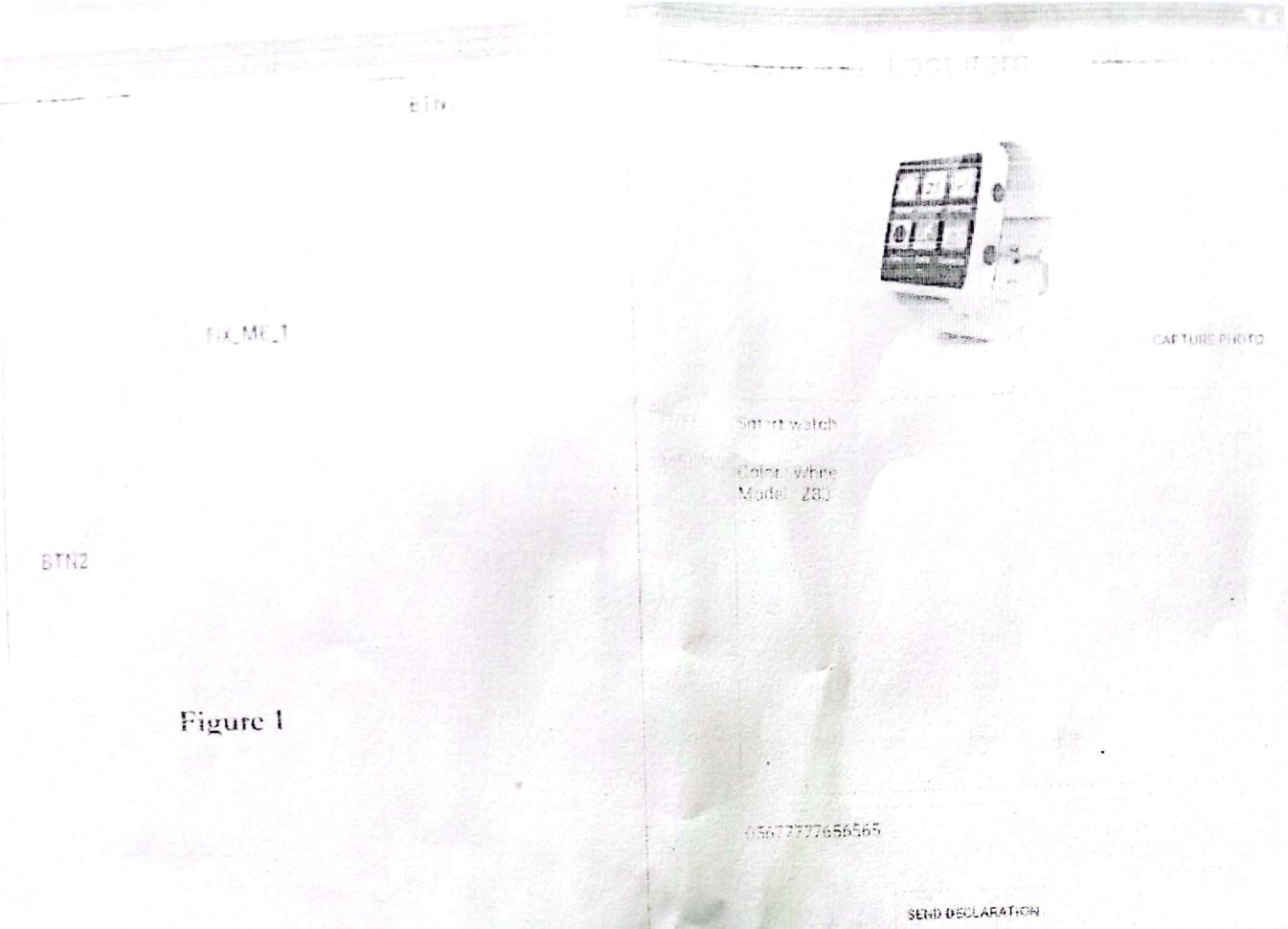
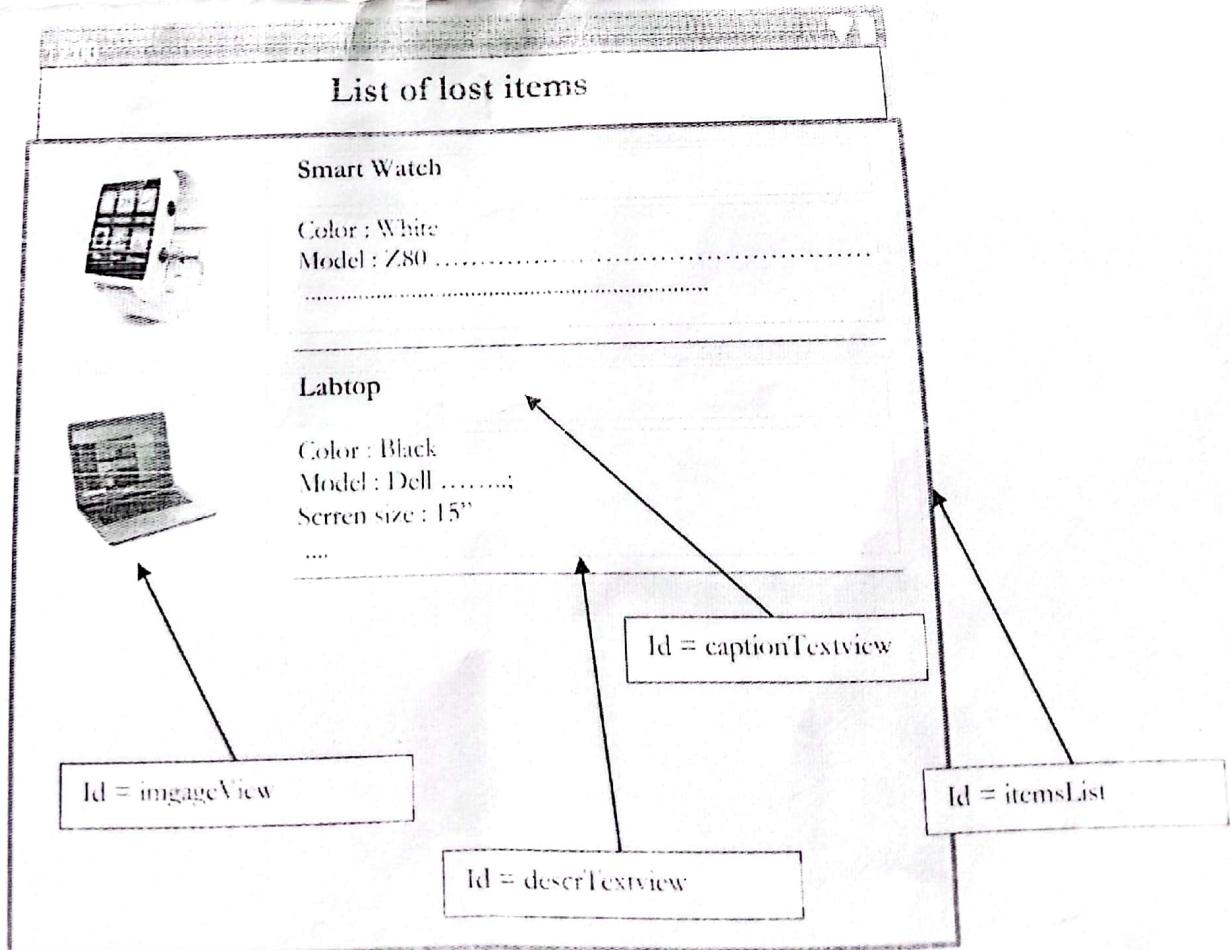


Figure 1

Figure 2

Figure 3



Implementation d'interface NoteView

Implementation de View pour NoteView

ArrayList<Module> noteModules

ArrayList<String> noteTitles

ArrayList<String> noteContent

ArrayList<String> noteType

### 3) Les entrées de la liste déroulante (4 points)

```
string-array name="nommoduledesitem"
    item<item>
        <item>Panneau solaire</item>
        <item>Panneau solaire</item>
        <item>Téléphone portable</item>
    </string-array>
```

### 4) La classe Module Adapter

```
public class ModuleAdapter extends ArrayAdapter {
    private final Context context;
    private final ArrayList<Module> list;

    public ModuleAdapter(Context context, ArrayList<Module> list) {
        super(context, R.layout.item, list);
        this.context = context;
        this.list = list;
    }

    @Override
    public int getCount() {
        return list.size();
    }

    @Override
    public long getItemId(int position) {
        return list.get(position).getId();
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflator = LayoutInflater.from(context);
        View v = inflator.inflate(R.layout.item, null);

        TextView moduleEntry = v.findViewById(R.id.moduleTv);
        TextView noteTv = v.findViewById(R.id.noteTv);

        moduleEntry.setText(list.get(position).getModule());
        noteTv.setText(list.get(position).getNote() + "\n");

        return v;
    }
}
```

```
    android:orientation="horizontal">"

    TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Année d'étude" />

    Spinner
        android:id="@+id/spinner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="3"
        android:entries="@array/annees_etude" />

</LinearLayout>

<Button
    android:id="@+id/button2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Afficher" />

<ListView
    android:id="@+id/listV"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:entries="@array/liste" />
</LinearLayout>
```

## 2) La classe Module (2points)

```
public class Module {
    String module;
    double note;

    public Module(String module, double note) {
        this.module = module;
        this.note = note;
    }

    public String getModule() {
        return module;
    }

    public void setModule(String module) {
        this.module = module;
    }

    public double getNote() {
        return note;
    }

    public void setNote(double note) {
        this.note = note;
    }
}
```