Exercice 2.3

Observez que la classe Etudiant a un constructeur avec 5 paramètres. Bien entendu, le nombre de paramètres risque de croître, puisque beaucoup d'autres attributs sont susceptibles d'être ajoutés à la classe Etudiant. Observez aussi que lorsqu'on construit une instance d'Etudiant il est facile de se tromper dans l'ordre des paramètres du constructeur. Heureusement que l'IDE vous aide en vous suggérant cet ordre lorsque vous programmez... De plus, les valeurs de certains attributs peuvent être inconnues au moment de la construction de l'objet : peut-être l'adresse de l'étudiant n'est pas encore connue, car l'attribution des logements universitaires par les organismes correspondants n'a pas encore eu lieu ; l'adresse mail n'est pas encore active au moment de l'inscription de l'étudiant, etc. Plusieurs solutions peuvent être envisagées :

1. Une solution est de définir plusieurs constructeurs avec différents paramètres et de les faire collaborer (voir les exemples en cours). Ainsi, l'utilisateur pourra choisir le constructeur qui lui convient. C'est ce qu'on appelle une *construction télescopique*. Essayons !
   Créez une nouvelle classe EtudiantTelescopique (en copiant les attributs de la classe Etudiant de l'exercice 2) et modifiez-la afin de pouvoir instancier les étudiants de 3 manières suivantes :
   - en indiquant uniquement le nom et le prénom
   - en indiquant uniquement le nom, le prénom et la date de naissance
   - en indiquant uniquement le nom, le prénom et le mail
2. Vérifiez votre programme dans la classe principale.
3. Avec cette approche, est-il possible d'ajouter un constructeur supplémentaire afin d'indiquer uniquement le nom, le prénom et l'adresse ? Pourquoi ?

Une autre solution serait de suivre le modèle *Java Beans*, en proposant un seul constructeur minimal et en fournissant des méthodes modifieurs (*setters*) pour chaque attribut de la classe. Créez une classe appelée EtudiantJavaBeans. Voici comment se fera alors la construction des objets de ce type :

```
class GestionEtudiants {

  public static void main(String[] args) {
    EtudiantJavaBeans toto = new EtudiantJavaBeans();
    toto.setNom("BenAli");
    toto.setDateDeNaissance(LocalDate.of(2003, Month.JANUARY, 28));
    /* ... */
    toto.setAdresse("99, av. Zaatcha, 07000 Biskra");
  }
        }
```

4. Comparez cette solution avec celle de la classe EtudiantTelescopique. Quels sont les avantages et les inconvénients ?

Bonus : Finalement, on vous demande de développer une solution en combinant les bonnes idées des deux versions précédentes. Voici comment on voudrait pouvoir créer des étudiants dans la classe principale :

```
class GestionEtudiants {
  public static void main(String[] args) {
    Etudiant lolo = new EtudiantBuilder()
      .ajouterNom("BenAli")
      .ajouterPrenom("Mohamed")
      .ajouterDateNaissance(LocalDate.of(2003, Month.JANUARY, 28))
      .ajouterMail("name@univ-biskra.dz")
```

```
        .ajouterAdresse("99, av. Zaatcha, 07000 Biskra")
        .build();
}
```

}

Remarquez qu'il n'y a qu'un seul symbole ; dans le programme ci-dessus. Autrement dit, l'instanciation de l'objet se fait avec une seule instruction, qui a été écrite sur plusieurs lignes pour plus de lisibilité.

Ajoutez la classe EtudiantBuilder à votre application pour que l'instruction ci-dessus fonctionne (l'exemple ci-dessous montre comment procéder).

Example:

```
public class User
{
        //All final attributes
        private final String firstName; // required
        private final String lastName; // required
        private final int age; // optional
        private final String phone; // optional
        private final String address; // optional

        private User(UserBuilder builder) {
                this.firstName = builder.firstName;
                this.lastName = builder.lastName;
                this.age = builder.age;
                this.phone = builder.phone;
                this.address = builder.address;
        }
public static class UserBuilder
        {
                private final String firstName;
                private final String lastName;
                private int age;
                private String phone;
                private String address;

                public UserBuilder(String firstName, String lastName) {
                        this.firstName = firstName;
                        this.lastName = lastName;
                }
                public UserBuilder age(int age) {
                        this.age = age;
                        return this;
                }
                public UserBuilder phone(String phone) {
                        this.phone = phone;
                        return this;
                }
                public UserBuilder address(String address) {
                        this.address = address;
                        return this;
                }
                //Return the finally consrcuted User object
                public User build() {
```

```
                User user =  new User(this);
                validateUserObject(user);
                return user;
        }
        private void validateUserObject(User user) {
                //Do some basic validations to check
                //if user object does not break any assumption of system
        }
    }
}
```

Then, we can use it like this:

```
public static void main(String[] args)
{
    User user1 = new User.UserBuilder("Lokesh", "Gupta")
    .age(30)
    .phone("1234567")
    .address("Fake address 1234")
    .build();

    System.out.println(user1);

    User user2 = new User.UserBuilder("Jack", "Reacher")
    .age(40)
    .phone("5655")
    //no address
    .build();

    System.out.println(user2);

    User user3 = new User.UserBuilder("Super", "Man")
    //No age
    //No phone
    //no address
    .build();

    System.out.println(user3);
}
```

Pour aller plus loin : Vous noterez que la classe EtudiantBuilder sert uniquement à instancier des objets Etudiant de manière "organisée" et lisible. Le problème est qu'il est toujours possible d'instancier un objet de type Etudiant sans utiliser le *builder* que vous avez écrit... C'est pour cela qu'il est possible d'améliorer la dernière solution en déclarant la classe EtudiantBuilder comme classe interne statique de la classe Etudiant et de rendre private le constructeur de la classe Etudiant. Ainsi la construction pourra se faire exclusivement à travers le *builder* et celui-ci servira uniquement à la construction des objets de type Etudiant, car c'est son seul rôle. Pour plus de détails et explications, voir le modèle de conception *Builder*.