FACULTE DES SCIENCES BEN M'SICK
UNIVERSITÉ HASSAN II DE CASABLANCA

Examen Java Session de Printemps 2025
SMI S6
Pr. Abdessamad Belangour

Un développeur freelance est un professionnel expérimenté et autonome qui exerce son activité en tant qu'indépendant. Dans le cadre de cet examen, il souhaite concevoir une application personnelle lui permettant de gérer efficacement ses missions de développement réalisées pour ses clients.

Chaque mission peut être facturée selon deux modes :

- **Au livrable** : un montant fixe est défini à l'avance, indépendamment de la durée ou du nombre de jours travaillés.
- **À la journée** : la facturation est basée sur un **taux journalier** appliqué au nombre de jours effectivement travaillés.

Les missions comportent plusieurs informations essentielles pour le suivi :

- un **statut** (en attente, en cours, terminée),
- une **date de début**,
- une **date de fin estimée**,
- ainsi que d'autres **informations complémentaires**.

Par ailleurs, le développeur collabore avec **plusieurs clients** et souhaite pouvoir **organiser et regrouper les missions par client** dans l'application.

Le code de l'application est comme suit :

```
public class Client {
    private String nom;
    /* on suppose que les constructeurs, les getters et setters, toString() et equals()
       sont fournis */
}
```

```
public enum Statut { EN_ATTENTE, EN_COURS, TERMINEE }
```

```
public class DateInvalideException extends Exception {
    public DateInvalideException(String message) { super(message);  }
}
```

```
import java.time.LocalDate;

public abstract class Mission implements Comparable<Mission> {
    protected String intitule;
    protected LocalDate dateDebut;
    protected LocalDate dateFinEstimee;
    protected Statut statut;
    protected Client client;

 public Mission(String intitule, LocalDate dateDebut, LocalDate dateFinEstimee,
Statut statut, Client client) throws DateInvalideException {
        ..........
        this.intitule = intitule;
        this.dateDebut = dateDebut;
        this.dateFinEstimee = dateFinEstimee;
        this.statut = statut;
        this.client = client;
    }
    public abstract double calculerMontant();
    public LocalDate getDateDebut() { return dateDebut; }
    public Client getClient() { return client; }
    @Override
    public int compareTo(Mission autre) {.....}
    @Override
    public String toString() { return intitule + " (" + statut + ") - Client: " + client; }
}
```

```
import java.time.LocalDate;

public class MissionFactureeAuLivrable extends Mission {
    private double montantFixe;
    public MissionFactureeAuLivrable(String intitule, LocalDate debut, LocalDate fin,
Statut statut, Client client, double montantFixe)  throws DateInvalideException {
        ..........
        this.montantFixe = montantFixe;
    }
    @Override
    public double calculerMontant() {  return montantFixe; }
}
```

```
import java.time.LocalDate;
public class MissionFactureeParJour extends Mission {
    private double tauxJournalier;
    private int nbJoursPrevus;
    public MissionFactureeParJour(String intitule, LocalDate debut, LocalDate fin,
Statut statut, Client client, double taux, int nbJours)
            throws DateInvalideException {   }

    @Override
    public double calculerMontant() {.....  }
}
```

```
import java.util.*;
import java.util.stream.*;

public class Freelance {
    private String nom;
    private Map<Client, List<Mission>> missionsParClient = new HashMap<>();

    public Freelance(String nom) {
        this.nom = nom;
    }

    public void ajouterMission(Mission m) {..... }

    public void supprimerMissionsTerminees() {..... }

    public double getTotalRevenu() {..... }

    public double getMontantFactureParClient(Client client) {.... }

    public List<Mission> getMissionsEnCours() {.... }

    public List<Mission> getMissionsTrieesParMontant() {.....}

    public Map<Client, Double> getRevenuParClient() { }

    public Optional<Client> getClientPlusRentable() {  }

    public void afficherTypeDeMission() {..... }
}
```

Questions :

- Dans la classe **Mission** :
    1) Donner le code manquant du **constructeur**
    2) Donner le code de la méthode **compareTo**
- Dans la classe **MissionFactureeAuLivrable**
    3) Donner le code manquant du **constructeur**
- Dans la classe **MissionFactureeParJour**
    4) Donner le code de la méthode **calculerMontant**
- Dans la classe **Freelance**, donner le code :
    5) De la méthode **ajouterMission**
    6) De la méthode **supprimerMissionsTerminees**
    7) De la méthode **getTotalRevenu** en passant par les **streams**
    8) De la méthode **getMontantFactureParClient**
    9) De la méthode **getMissionsEnCours**
    10) De la méthode **getMissionsTrieesParMontant**
    11) De la méthode **afficherTypeDeMission**