BUT 1 2025-2026

Développement orienté objet - Java - TP noté Février 2026. Durée 2 heures.

Pour cet examen, votre fond d'écran doit être vert clair. Vous devez obligatoirement placer tous les fichiers que vous souhaitez rendre dans le répertoire `EXAM`, qui est déjà présent sur votre compte à l'ouverture de la session. Tout ce qui ne se trouve pas dans ce dossier sera perdu lorsque vous vous déconnecterez. Ne créez pas vous-même le répertoire `EXAM`. Placez tous vos fichiers dans celui qui existe déjà.

Sous Eclipse, votre workspace doit être dans `EXAM`. Sinon, configurez-le (via File > Switch Workspace) pour qu'il corresponde à un répertoire dans `EXAM`. Attention : les noms des classes, des champs et des méthodes demandés doivent impérativement être respectés. De plus, le code doit être correctement indenté.

Commencez par vérifier la configuration d'Eclipse.

- Vérifiez que le JRE est /usr/local/apps/java25.
- Vérifiez que le compilateur est java 25
- Créez ensuite votre Java project qui portera votre nom : `NOM_PRENOM`.

La javadoc 25 et les slides du cours (seuls documents autorisés) sont accessibles ici :
`https://ligm.univ-mlv.fr/~juge/javadoc-25/`
`https://ligm.univ-mlv.fr/~beal/JavaBUT1.html`

Les tests des exemples donnés dans chaque question doivent être réalisés (les mettre en commentaire s'ils ne fonctionnent pas). Ces tests sont pris en compte dans la notation. **L'utilisation de `instanceof` est interdite.** Vous n'aurez pas à écrire les méthodes `equals` et `hashCode`.

Dans la première partie, toutes les classes seront écrites dans le package `fr.uge.spacetravel`. Dans la deuxième partie, toutes les classes seront écrites dans le package `fr.uge.spacetravel2`. Vous écrirez une méthode `main` dans une classe `Test` dans chaque package pour effectuer les tests.

Dans cet examen, on modélise des vaisseaux spatiaux qui peuvent embarquer des passagers.

Partie 1

Exercice 1.

1. Membre d'équipage (`Crew`)

   Un membre d'équipage (`Crew`) est caractérisé par

   - son nom (`name`) (une chaîne de caractères),
   - son âge (`age`, un nombre entier d'années) qui ne peut pas dépasser 60 ans,
   - son grade de type `String`

   Écrire un record `Crew` permettant de créer et d'afficher un objet de type `Crew` possédant ces trois champs. L'affichage devra être de la forme :

   ```
   IO.println("test 1");
   var crew = new Crew("James", 40, "ROOKIE");
   IO.println(crew);
   ```

   Sortie attendue :

   ```
   Crew James (40 years ROOKIE)
   ```

2. Vaisseau spatial (`SpaceShip`)

   Créer une classe pour modéliser un vaisseau spatial `SpaceShip` contenant :

- un nom (`name` de type `String`)
- une liste de membres d'équipage (`passengers`, de type `ArrayList<Crew>`).

Écrivez une méthode `add` permettant d'ajouter un membre d'équipage au vaisseau. On ne vérifiera pas si un membre d'équipage est déjà présent avant de l'ajouter (un même membre d'équipage pourra donc figurer plusieurs fois dans la liste).

3. Affichage du vaisseau

   Ajoutez une méthode `toString` pour afficher le vaisseau en commençant par son nom, suivi des membres d'équipage (un par ligne) :

```
IO.println("test 2");
var artemis = new SpaceShip("Artemis");
artemis.add(new Crew("James", 40, "ROOKIE"));
artemis.add(new Crew("Naomi", 50, "STAFF"));
artemis.add(new Crew("Amos", 45, "OFFICER"));
artemis.add(new Crew("Camina", 30, "OFFICER"));
IO.println(artemis);
```

   Sortie attendue :

```
Artemis
Crew James (40 years ROOKIE)
Crew Naomi (50 years STAFF)
Crew Amos (45 years OFFICER)
Crew Camina (30 years OFFICER)
```

4. Méthode hibernation dans `Crew`

   Ajoutez une méthode `int hibernation()` dans le record `Crew` qui calcule le temps maximal d'hibernation pour chaque membre d'équipage. Ce temps dépend de leur grade :

   - 10 ans pour un `"ROOKIE"`,
   - 15 ans pour un `"STAFF"`,
   - 30 ans pour un `"OFFICER"`.
   - 0 année pour tout autre grade.

5. Méthode hibernation dans `SpaceShip`

   Écrivez une méthode `int hibernation()` qui calcule le **minimum** des temps maximaux d'hibernation de tous les membres d'équipage. Une exception sera levée si le vaisseau est vide.

```
IO.println("test 3");
IO.println(artemis.hibernation());
```

   Sortie attendue :

```
test 3
10
```

Partie 2

Exercice 2.

Copiez-collez le package `fr.uge.spacetravel` dans `fr.uge.spacetravel2` et travaillez dans ce nouveau package. Gardez le record `Crew`. Videz le `main` de `Test`. Dans la classe `SpaceShip`, commentez pour l'instant la méthode `int hibernation`.

Dans cette partie, le vaisseau peut maintenant contenir deux types de passagers :

- des membres d'équipage (`Crew`),

- des touristes (`Tourist`).

1. Touriste (`Tourist`).

   Un touriste est caractérisé par :

   - son nom (`name`) (une chaîne de caractères),
   - un booléen (`vip`) indiquant s'il est VIP,
   - un nombre de crédits (`credits`, un entier positif ou nul au maximum 30 000).

   Écrire le code permettant de créer et afficher un objet de type `Tourist`.

```
IO.println("test 1");
var tourist = new Tourist("Jeff", true, 30_000);
IO.println(tourist);
var tourist2 = new Tourist("Elon", false, 10_000);
IO.println(tourist2);
```

   Sortie attendue :

```
test 1
Tourist Jeff (vip 30000)
Tourist Elon (10000)
```

2. Écrire le code d'une interface `Passenger` qui sera un type commun pour `Crew` et `Tourist`.

3. Modifiez la classe `SpaceShip` pour qu'elle contienne désormais une liste de `Passenger`. Modifiez la méthode `add` pour qu'elle permette d'ajouter aussi bien des membres d'équipage que des touristes.

```
IO.println("test 2");
var artemis = new SpaceShip("Artemis");
artemis.add(new Crew("Clervoy", 40, "ROOKIE"));
artemis.add(new Crew("Pesquet", 50, "STAFF"));
artemis.add(new Crew("Adenot", 45, "OFFICER"));
artemis.add(new Tourist("Jeff", true, 30_000));
artemis.add(new Tourist("Elon", false, 10_000));
artemis.add(new Crew("Baudry", 30, "OFFICER"));
IO.println(artemis);
var blueOrigin = new SpaceShip("Blue Origin");
blueOrigin.add(new Tourist("Jeff", true, 30_000));
blueOrigin.add(new Tourist("Elon", false, 10_000));
IO.println(blueOrigin);
```

   Sortie attendue :

```
test 2
Artemis
Crew Clervoy (40 years ROOKIE)
Crew Pesquet (50 years STAFF)
Crew Adenot (45 years OFFICER)
Tourist Jeff (vip 30000)
Tourist Elon (10000)
Crew Baudry (30 years OFFICER)
Blue Origin
Tourist Jeff (vip 30000)
Tourist Elon (10000)
```

4. Ajoutez dans le record `Tourist` une méthode `int hibernation()` qui calcule le temps maximal d'hibernation pour un touriste. Le temps maximal d'hibernation pour un touriste est :

- pour les touristes VIP, 60 ans,
- pour les touristes non VIP, 50 ans.

5. Modifiez le code pour que la méthode `int hibernation()` de `SpaceShip` fonctionne. Cette méthode calcule maintenant le minimum des temps maximaux d'hibernation de tous les passagers. Une exception sera levée si le vaisseau est vide.

```
IO.println("test 3");
IO.println(artemis.hibernation());
IO.println(blueOrigin.hibernation());
```

Sortie attendue :

```
test 3
10
50
```

6. Écrire une méthode `List<Passenger> tourists()` qui renvoie la liste des passagers du vaisseaux qui sont des touristes.

```
IO.println("test 4");
IO.println(artemis.tourists());
```

Sortie attendue :

```
[Tourist Jeff (vip 30000), Tourist Elon (10000)]
```