BUT 1 2025-2026

Développement orienté objet - Java - TP noté 2 Avril 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-eiffel.fr/~juge/javadoc-25/`
`https://ligm.univ-eiffel.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.ferry`. Dans la deuxième partie, toutes les classes seront écrites dans le package `fr.uge.ferry2`. Vous écrirez une méthode `main` dans une classe `Main` dans chaque package pour effectuer les tests.

Dans cet examen, on modélise des ferries qui embarquent des voitures ou des camions à bord.

Partie 1

Exercice 1.

1. Personne (`Person`)

   Une personne (`Person`) est caractérisée par

   - son nom (`name`) (une chaîne de caractères),
   - son âge (`age`, un nombre entier d'années) qui est positif ou nul.

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

   ```
   var person = new Person("Rayu", 25);
   IO.println(person);
   ```

   Sortie attendue :

   ```
   Rayu 25
   ```

2. Voiture (`Car`)

   Une voiture (`Car`) est caractérisée par

   - son nombre de passager (`passengersNumber`, un nombre entier) qui est positif ou nul,
   - son nombre d'enfants (`kidsNumber`, un nombre entier) qui est positif ou nul,

- son propriétaire (`owner`) de type `Person`.

Le nombre d'enfants ne devra pas être supérieur au nombre de passagers moins 1.

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

```
var car = new Car(4, 0, new Person("Rayu", 25));
IO.println(car);
```

Sortie attendue :

```
Car Rayu 25 (4, 0)
```

3. Camion (`Truck`)

   Un camion (`Truck`) est caractérisé par

   - son poids (`weight`, un nombre entier) qui est positif ou nul,
   - son propriétaire (`owner`) de type `Person`.

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

```
var truck = new Truck(2000, new Person("Rayan", 40));
IO.println(truck);
```

   Sortie attendue :

```
Truck Rayan 40 (2000)
```

4. Ferry (`Ferry`) et interface `Vehicle`

   Un ferry `Ferry` contient :

   - une liste de véhicules (`ferry`), de type `ArrayList<Vehicle>`,

   où `Vehicle` sera un type commun pour les voitures et les camions qui peuvent embarquer sur le ferry.

   Écrivez l'interface `Vehicle` et la classe `Ferry`.

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

5. Affichage du ferry

   Ajoutez une méthode `toString` pour afficher le ferry en affichant un véhicule par ligne :

```
IO.println("test 1");
var v1 = new Car(4, 1, new Person("David", 35));
var v2 = new Car(4, 0, new Person("Rayu", 25));
var v3 = new Car(4, 2, new Person("Rayan", 40));
var v4 = new Truck(2000, new Person("Rayan", 40));
var v5 = new Truck(4000, new Person("Rayan", 40));
var f1 = new Ferry();
f1.add(v1);
f1.add(v2);
f1.add(v2);
f1.add(v3);
f1.add(v4);
f1.add(v5);
IO.println(f1);
```

   Sortie attendue :

```
test 1
Car David 35 (4, 1)
Car Rayu 25 (4, 0)
Car Rayu 25 (4, 0)
Car Rayan 40 (4, 2)
Truck Rayan 40 (2000)
Truck Rayan 40 (4000)
```

6. Méthode `fare` dans `Car`

   Ajoutez une méthode `public int fare()` dans le record `Car` qui renvoie le tarif de la traversée pour une voiture. Ce tarif est :

   - 20 multiplié par le nombre de passagers.

7. Méthode `fare` dans `Truck`

   Ajoutez une méthode `public int fare()` dans le record `Truck` qui renvoie le tarif de la traversée pour un camion.

   - le poids divisé par 5 (`weight / 5`, un entier)

8. Méthode `revenues` dans `Ferry`

   Écrivez une méthode `public int revenues()` qui calcule la somme acquittée par tous les véhicules pour la traversée, c'est-à-dire, la somme de toutes les valeurs de `fare()` pour chaque véhicule.

```
IO.println("test 2");
IO.println(f1.revenues());
```

   Sortie attendue :

```
test 2
1520
```

9. Méthode `cars` dans `Ferry`

   Écrivez une méthode `public List<Vehicle> cars()` qui renvoie une liste non modifiable de tous les véhicules du ferry qui sont des voitures.

```
IO.println("test 3");
IO.println(f1.cars());
```

   Sortie attendue :

```
test 3
[Car David 35 (4, 1), Car Rayu 25 (4, 0), Car Rayu 25 (4, 0), Car Rayan 40 (4, 2)]
```

10. Méthode `revenuesWithDiscount` dans `Ferry`

    Écrivez une méthode `public int revenuesWithDiscount(int discount)` dans `Ferry` qui calcule la somme acquittée par tous les véhicules pour la traversée en effectuant une réduction pour certaines voitures. On écrira pour cela, dans les records `Car` et `Truck`, des méthodes `public int fareWithDiscount(int discount)`. Pour un camion il n'y a pas de discount et pour une voiture le tarif avec discount est :

    - le tarif normal moins une réduction égale à `discount` multiplié par le nombre d'enfants.

    La valeur renvoyée par `fareWithDiscount(discount)` devra toujours être positive ou nulle. La réduction ne peut être supérieure au tarif normal (sinon renvoyez 0), et l'argument `discount` devra être positif ou nul (sinon levez une exception).

```
IO.println("test 4");
IO.println(f1.revenuesWithDiscount(10));
```

Sortie attendue :

```
test 4
1490
```

Partie 2

Exercice 2.

Copiez-collez le package `fr.uge.ferry` dans `fr.uge.ferry2` et travaillez dans ce nouveau package. Gardez les records. Vous pouvez garder le `main` en commentant `IO.println(f1.revenuesWithDiscount(10));`. Dans la classe `Ferry`, enlevez tout.

Dans cette partie, le ferry est implanté avec une map qui donne pour chaque personne, la liste des véhicules appartenant à cette personne qui sont à bord du ferry.

1. Ferry avec une `HashMap<Person, List<Vehicle>>`.

   Écrire la classe `Ferry` qui contient une map de type `HashMap<Person, List<Vehicle>>`.

2. Ajouter une méthode `add(Vehicle vehicle)` qui permet d'ajouter un véhicule dans le ferry. Si le propriétaire du véhicule est déjà une clé de la map, le véhicule s'ajoute à la fin de la liste des véhicules du propriétaire. Sinon, une nouvelle entrée de la map est créée avec, comme clé, le propriétaire, et comme valeur associée, une `ArrayList<Vehicle>` qui contient cet unique véhicule.

3. Ajoutez une méthode `toString` pour afficher le ferry comme ci-dessous :

```
IO.println("test 1");
var v1 = new Car(4, 1, new Person("David", 35));
var v2 = new Car(4, 0, new Person("Rayu", 25));
var v3 = new Car(4, 2, new Person("Rayan", 40));
var v4 = new Truck(2000, new Person("Rayan", 40));
var v5 = new Truck(4000, new Person("Rayan", 40));
var f1 = new Ferry();
f1.add(v1);
f1.add(v2);
f1.add(v2);
f1.add(v3);
f1.add(v4);
f1.add(v5);
IO.println(f1);
```

```
test 1
David 35: [Car David 35 (4, 1)]
Rayan 40: [Car Rayan 40 (4, 2), Truck Rayan 40 (2000), Truck Rayan 40 (4000)]
Rayu 25: [Car Rayu 25 (4, 0), Car Rayu 25 (4, 0)]
```

   Chaque entrée de la map est sur une ligne.

4. Écrire la méthode `public int revenues()` qui doit renvoyer la somme acquittée par tous les véhicules pour la traversée.

```
IO.println("test 2");
IO.println(f1.revenues());
```

```
test 2
1520
```

   On doit trouver le même résultat que pour la partie 1 bien sûr.