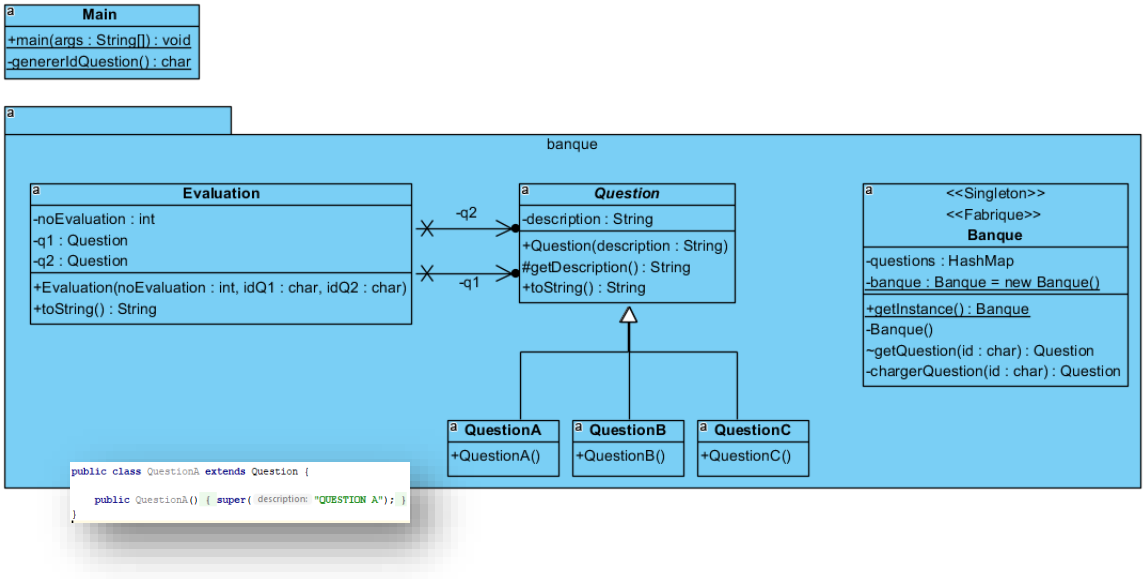
Patron #5 : Poids mouche (flyweight)



Contraintes

- Supposons une ressource extrêmement lourde en mémoire et longue à charger : les Questions !
- On veut générer des centaines d'évaluations qui utilisent aléatoirement les mêmes questions. On veut éviter les problèmes de lenteur !
- La classe « **Banque** »
  - Singleton : afin de contrôler le chargement des questions en un seul exemplaire en mémoire
  - Fabrique : afin de générer les questions. Elle ne connait pas les questions de l'examen alors elle les charge à la demande et les conserve pour les prochaines demandes. Dans mon cas, j'ai utilisé un HashMap (l'ID de la question est la clé) mais vous pouvez utiliser la méthode de votre choix (c'est de l'implémentation interne à votre classe)
  - Voici le code de ma méthode chargerQuestion (qui est appelée par getQuestion), ça pourrait vous orienter pour le chargement dynamique des classes en Java (oui, je sais, newInstance est deprecated)

```
private Question chargerQuestion(char id) {

    Question question = null;
    try {
        Class<Question> classeQuestion = (Class<Question>) Class.forName("banque.Question" + id);
        question = classeQuestion.newInstance();
        questions.put(id, question);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return question;
}
```

Exemple d'utilisation

- Avec des questions en doublons pour m'assurer de la compéhension des étudiants ;-)

```java
public static void main(String args[]){

    final int NB_ETUDIANTS = 149;
    ArrayList<Evaluation> mesEvaluationsAndroid = new ArrayList<Evaluation>();

    // creation des evaluations
    for (int x = 1; x <= NB_ETUDIANTS; x++)
        mesEvaluationsAndroid.add(new Evaluation(x, genererIdQuestion(), genererIdQuestion()));

    // Affichage des évaluations
    for (Evaluation evaluation : mesEvaluationsAndroid)
        System.out.print(evaluation);
}

private static char genererIdQuestion(){

    return (char)(ThreadLocalRandom.current().nextInt( origin: 65, bound: 68));
}
```

```
Evaluation #7
    q1 Question{QUESTION A}
    q2 Question{QUESTION C}
Evaluation #8
    q1 Question{QUESTION C}
    q2 Question{QUESTION B}
Evaluation #9
    q1 Question{QUESTION B}
    q2 Question{QUESTION C}
```