Programmation 1

Licence 1 – Portail René Descartes

2018 – 2019

Aix*Marseille université
Initiative d'excellence

Première partie I

Introduction générale

Aix*Marseille université Initiative d'excellence

Programmation 1 : organisation

Cours obligatoire pour tous :

- 12 semaines, comprenant chacune :
- 1h30 de cours magistral (CM),
- 1h30 de travaux dirigés (TD),
- 2h de travaux pratiques sur ordinateur (TP).

Emploi du temps sur ADE.

Évaluation :

- session 1 : travaux pratiques sur 8 points, examen final sur 12 points.
- session 2 : examen final sur 20 points.

Aix*Marseille université
Initiative d'excellence

Programmation 1 : enseignants

Enseignants pour Luminy :

- CM : Guyslain Naves,
- TD : Yassine Motie, Guyslain Naves, Karim Nouioua,
- TP : Jérémy Auguste, Sébastien Delecraz, Thibault Magallon, Yassine Motie, Guyslain Naves.

Contact : courriel `prénom.nom@univ-amu.fr`

Aix*Marseille université
Initiative d'excellence

Objectifs du cours

À la fin du cours vous devrez être capable de :

- lire, écrire, modifier des programmes dans un langage de programmation utilisé par l'industrie du logiciel (Java),
- vous servir de bibliothèques de programmes en consultant leur documentation,
- trouver et corriger une erreur dans un programme,
- ajouter une fonctionnalité à un programme,
- respecter des bonnes pratiques de programmation,
- comprendre les enjeux liés aux exigences de qualité des programmes.

Programme du cours

- Concepts de base en programmation : programme, compilation, instructions et expressions, types, instruction conditionnelle.
- Concepts de base en programmation orientée objet : objets, méthodes et propriétés, classes.
- Concepts avancées en programmation : collection, itération, utilisation de bibliothèques.
- Concepts avancées en programmation orientée objet : interfaces.
- Pratique de la programmation : refactorisation, tests, détection et corrections des erreurs.

Aix Marseille université
Initiative d'excellence

Qu'est-ce que la programmation ?

La programmation

C'est l'activité consistant à anticiper, planifier, organiser la résolution d'un ensemble de tâches, dans un langage compréhensible par l'humain et la machine, afin de fournir un service ou de répondre à une besoin.

Aix*Marseille université
Initiative d'excellence

Fonctionnement d'un programme typique

1. L'environnement ou un utilisateur produit des stimuli, via un *périphérique* (clavier, souris, carte réseau, capteurs, mémoire, un autre programme). Ce sont les *entrées* du programme.
2. Ces stimuli sont numérisés, représentés par du texte.
3. Le programme reçoit et analyse ces données textuelles, en utilisant des représentations structurées, et produit des résultats représentés par du texte : les *sorties*.
4. Les sorties sont envoyées sur les périphériques, pour provoquer un affichage, un son, un processus mécanique ou l'envoi d'un message sur un réseau, ou pour être stockées.

Aix*Marseille université
Initiative d'excellence

Programmer : une activité de long terme.

Étapes du développement d'un programme :

- Analyser les besoins.
- Spécifier les comportements du programme.
- Concevoir des méthodes de résolution (des *algorithmes*).
- Implémenter le programme (vulgairement : *coder*).
- Vérifier le bon comportement du programme (*tester*).
- Déployer le programme dans son environnement.
- Maintenir le programme (corriger des défaillances, ajouter des fonctionnalités).

Aix*Marseille université
Initiative d'excellence

Une multitude de formes

Exemples de programmes :

- lecteur média,
- navigateur Web,
- application de smartphone,
- contrôleur de production industrielle,
- service web,
- système d'exploitation,
- compilateur, . . .

Aix*Marseille université
Initiative d'excellence

Une multitude de langages

Différents langages existent pour répondre à tous les besoins, voici quelques traits de langages :

- procédural (séquence d'instructions),
- déclaratif (définitions de valeurs),
- logique (collection de contraintes),
- fonctionnel (composition de fonctions),
- objet (composants en interaction),

Aix*Marseille université
Initiative d'excellence

Une cible : le processeur

Vocation d'un programme : **être exécuté par une machine**.

- toutes les machines (téléphone, ordinateur, . . .) sont basées sur des principes similaires,
- (au moins) un processeur réalise une liste d'*instructions*,
- chaque instruction provoque un changement minime de l'état de la mémoire de la machine,
- chaque instruction est réalisée extrêmement rapidement (quelques nanosecondes).

Langages : Langage machine (instructions encodées par des entiers), assembleur (instructions encodées par des mots)

Aix*Marseille université Initiative d'excellence

Limitation de l'assembleur

Limitation : très peu d'instructions (quelques dizaines) !

Illustration : utiliser uniquement les 100 mots les plus courants de la langue française :

> *être avoir dans elle pour vous faire plus dire nous comme mais pouvoir avec tout aller voir bien sans leur homme deux mari vouloir femme venir quand grand celui notre devoir jour prendre même votre tout rien petit encore aussi quelque dont tout trouver donner temps même falloir sous parler alors main chose mettre savoir yeux passer autre après regarder toujours puis jamais cela aimer heure croire cent monde donc enfant fois seul autre entre vers chez demander jeune jusque très moment rester répondre tout tête père fille mille premier entendre trois cœur ainsi quatre terre contre dieu monsieur voix*

Aix*Marseille université
Initiative d'excellence

Langage de programmation haut-niveau

Principes d'un langage haut-niveau :

- donner un vocabulaire suffisant pour exprimer toute tâche réalisable,
- mais raisonnablement simple et universel pour être facile à apprendre, comprendre et partager,
- donner un mécanisme permettant d'enrichir le vocabulaire, de définir de nouveaux mots spécifiques à des contextes précis ou à une tâche à réaliser,
- les nouveaux mots peuvent être des verbes (des actions, des fonctions) ou des noms (des valeurs),
- fournir un outil convertissant le programme en assembleur ou langage machine (afin d'*exécuter* le programme) : le *compilateur*.

Composants d'un langage de programmation

Un langage définit donc :

- une **syntaxe** : caractères valides, ponctuation, espacement, majuscule, ...
- un **vocabulaire** : des noms désignant des entités ou des valeurs, des verbes désignant des actions ou des manipulations,
- une **grammaire** : des règles spécifiant l'organisation des mots en phrases porteuse de sens (avec sujet, verbe, compléments par exemple),
- une **sémantique** : une signification associée à chaque phrase ou texte grammaticalement correct.

.

Aix*Marseille université Initiative d'excellence

Syntaxe (illustration en Java)

```
public static int greatestCommonDivisor(int n, int d) {
  if (d == 0) { return n; }
  while (n > d) {
    n = n - d;
  }
  return greatestCommonDivisor(d, n);
}
```

Syntaxe :

- { et } pour grouper des instructions,
- (et) pour délimiter une condition,
- ; pour terminer des instructions,
- identifiants en caractères alphabétiques,
- espacement, indentation, retour à la ligne, . . .

Aix*Marseille université
Initiative d'excellence

Vocabulaire (illustration en Java)

```java
public static int greatestCommonDivisor(int n, int d) {
  if (d == 0) { return n; }
  while (n > d) {
    n = n - d;
  }
  return greatestCommonDivisor(d, n);
}
```

Vocabulaire :

- `greatestCommonDivisor` est un nouveau mot défini par ce fragment,
- `public`, `static`, `int`, `if`, `return`, `while` sont des mots de base du langage,
- `n` et `d` sont des mots propres à ce fragment de programme,
- `<`, `==`, `-` sont aussi des mots de base du langage.

Aix*Marseille université Initiative d'excellence

Grammaire (illustration en Java)

```
public static int greatestCommonDivisor(int n, int d) {
  if (d == 0) { return n; }
  while (n > d) {
    n = n - d;
  }
  return greatestCommonDivisor(d, n);
}
```

Grammaire :

- `public static int` pour déclarer une nouvelle entité,
- `if (condition) { instructions }` pour réaliser des instructions seulement si une condition est valide,
- `return expression;` pour terminer le calcul, ...

Aix*Marseille université
Initiative d'excellence

Sémantique (illustration en Java)

```java
public static int greatestCommonDivisor(int n, int d) {
  if (d == 0) { return n; }
  while (n > d) {
    n = n - d;
  }
  return greatestCommonDivisor(d, n);
}
```

Sémantique :

- calcul du plus grand diviseur commun de deux entiers,
- par l'algorithme d'Euclide,
- spécifiquement par soustractions successives.

Aix*Marseille université
Initiative d'excellence

Programmer pour la machine

Le compilateur qui traduit le programme en instructions processeurs n'est pas intelligent :

- il ne peut pas comprendre l'**intention** du programmeur,
- il exige un **respect strict** de la syntaxe, du vocabulaire, de la grammaire,
- il impose tout aussi strictement **sa** sémantique.

Ainsi :

- une **erreur** de **syntaxe**, de **vocabulaire** ou de **grammaire** empêche la compilation,
- une **mésentente** entre l'**intention** du programmeur et l'**interprétation** du compilateur provoque des comportements inattendus et faux à l'exécution du programme.

Aix*Marseille université Initiative d'excellence

Programmer pour les humains

Le programme est aussi écrit **à l'intention des humains** :

- il est écrit, modifié, corrigé par des humains,
- il documente une technique précise de résolution d'une tâche (exemple : source d'un programme comme *parcoursup* utilisé par un gouvernement), c'est un **document de référence**,
- il doit être convaincant.

Ainsi un bon programme :

- est clair, facile à lire, facile à modifier,
- est aussi simple que possible,
- respecte les conventions de style en vigueur,
- utilise un vocabulaire adapté à la tâche à réaliser.

Aix*Marseille université Initiative d'excellence

Organisation d'un programme

Un **programme** est composé de fichiers de texte (au moins un) : les *sources* du programme.

Chaque fichier source contient :

- des définitions de mots,
- et/ou une description de la réalisation d'une tâche par le programme.

Le texte d'un fichier source :

- est régi par la **grammaire** et la **syntaxe** propre au langage de programmation,
- ne doit pas avoir de styles (pas de couleurs, de polices,. . .)

Organisation d'un programme

Tout programme sérieux est composé de plusieurs fichiers, dont :

- des sources (le programme lui-même),
- des fichiers de tests,
- des fichiers de configurations,
- des ressources (images, données, ...).

Ceci constitue un **projet**.

Un projet est en général enregistré dans un répertoire propre, comprenant plusieurs sous-répertoires selon la nature des fichiers (sources, tests, fichiers compilés, ressources, ...)

Aix*Marseille université Initiative d'excellence

Compilateur

Compilateur

Un compilateur est un logiciel transformant des fichiers sources depuis un langage vers un autre langage (typiquement de l'assembleur ou du code machine)

Interpréteur

Un interpréteur est un logiciel qui exécute directement un programme donné par ses fichiers sources.

En Java :

- sources `.java` compilés par *javac* en `.class`,
- fichiers `.class` interprétés par *java*.

Aix*Marseille université Initiative d'excellence

Environnement de développement

Environnement intégré de développement (IDE)

Un IDE est un logiciel qui permet l'édition des fichiers sources d'un programme, en procurant des facilités pour la programmation.

- formattage du programme,
- compilation, exécution, affichage des erreurs,
- outils de détection des erreurs,
- intégration de la documentation, ...

Exemples : *IntelliJ*, *Eclipse*, *PyCharm*, *vi*, *emacs*, *Visual Studio Code*, ...

Aix*Marseille université
Initiative d'excellence

Deuxième partie II

Introduction à la programmation

Aix*Marseille université
Initiative d'excellence

La programmation objet

Les objectifs :

- faciliter le développement et l'évolution des programmes,
- permettre le travail en équipe,
- augmenter la qualité des logiciels.

Comment ?

- découpler (séparer) les parties du programme,
- limiter et localiser les modifications lors des évolutions du programme,
- permettre de réutiliser les composants du programme.

Aix*Marseille université Initiative d'excellence

Le langage Java

Le langage utilisé dans ce cours est Java :

- langage orienté objet,
- robuste et sûr, fonctionnant sur toute plateforme,
- performant,
- très populaire dans l'industrie,
- disposant d'excellents outils et bibliothèques.

Java intègre les mêmes concepts que la plupart des langages de programmation.

Un exemple d'application

On souhaite réaliser une application permettant d'affecter des étudiants à des cours d'option :

- chaque cours a une capacité maximale,
- chaque étudiant a une liste de préférences sur les cours qu'il souhaite suivre,
- chaque étudiant doit s'inscrire à un cours,
- la liste des cours, la liste des étudiants et de leurs préférences sont disponibles sous la forme de fichiers de données,
- nous souhaitons obtenir un fichier de données précisant pour chaque cours la liste des étudiants inscrits.

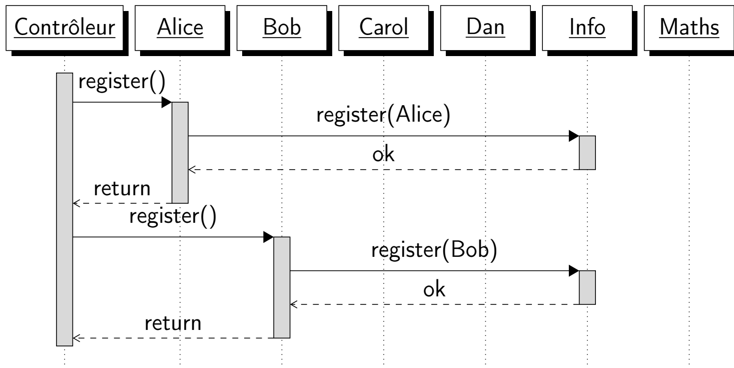
Les éléments de cette tâche

- des étudiants,
- des cours,
- des listes de préférences,
- des listes d'inscrits,
- des fichiers de données à lire ou écrire.

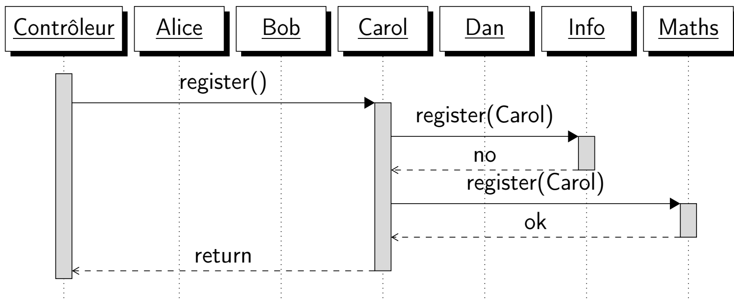
Chaque élément va être représenté comme un objet. Chaque objet possède ses propres instructions qui régissent son comportement.

Aix*Marseille université
Initiative d'excellence

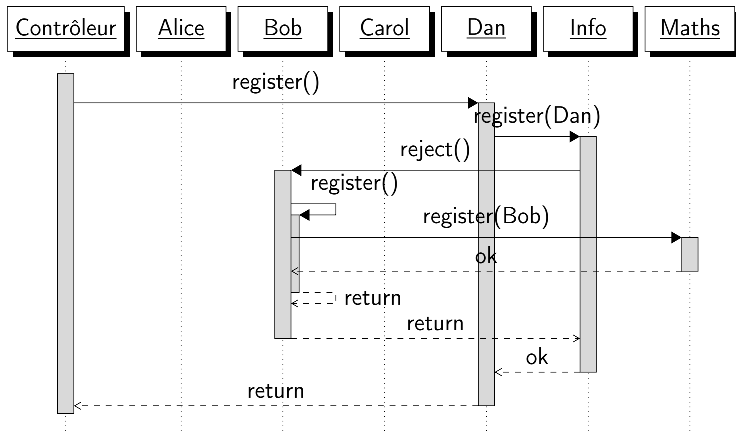
Exemple de processus



Exemple de processus



Exemple de processus



Comment maîtriser cette complexité ?

En programmation objet :

- Le comportement du programme est le résultat de la communication des objets entre eux.
- Les interactions produites sont en général trop complexes pour être appréhendées par un être humain.

Mais on réfléchit uniquement au niveau de l'objet :

- Chaque objet doit avoir un seul rôle, une seule responsabilité.
- Chaque objet doit avoir un comportement simple, facilement compréhensible par l'humain.
- Chaque objet doit limiter ses interactions avec les autres objets au nécessaire pour remplir son rôle.

Un étudiant

Un étudiant possède :

- un nom,
- une liste de préférences de cours.

Et il peut :

- s'inscrire à un cours,
- se faire désinscrire d'un cours.

Aix*Marseille université
Initiative d'excellence

Un cours

Un cours possède :

- une capacité,
- une liste d'étudiants inscrits.

Et il peut :

- traiter l'inscription d'un étudiant,
- fournir la liste des étudiants inscrits.

Aix Marseille université
Initiative d'excellence

Une liste de préférences

Une liste de préférences de cours possède :

- une séquence ordonnée de cours.

Et elle peut :

- fournir le prochain cours dans la liste,
- déclarer qu'il n'y a plus de cours sur la liste.

Aix*Marseille université Initiative d'excellence

Les objets

Étudiants, cours et listes de préférences ont donc :

- des propriétés (le nom de l'étudiant, la capacité du cours, la séquence de cours de la liste de préférences, ...),
- des comportements (s'inscrire à un cours, fournir la liste des étudiants inscrits, fournir le prochain cours dans l'ordre de préférence).

Un objet

Un objet est une entité composée de propriétés (ou champs ou *attributs*), et de comportements (ou méthodes ou service).

Aix*Marseille université
Initiative d'excellence

L'étudiante Alice

Propriétés :

- **nom** : "Alice"
- **préférences** : Informatique, Mathématiques, Physique, Mécanique.

Comportements :

- **s'inscrire** : s'inscrit au premier cours de sa liste de préférences duquel elle n'a pas encore été rejetée, jusqu'à ce que son inscription soit acceptée (ou avoir épuisé tous les cours).
- **se faire rejeter d'un cours** : s'inscrire.

Aix*Marseille université
Initiative d'excellence

L'étudiant Bob

Propriétés :

- **nom :** "Bob"
- **préférences :** Informatique, Mécanique, Physique, Mathématiques.

Comportements :

- **s'inscrire :** s'inscrit au premier cours de sa liste de préférences duquel il n'a pas encore été rejeté, jusqu'à ce que son inscription soit acceptée (ou avoir épuisé tous les cours).
- **se faire rejeter d'un cours :** s'inscrire.

Aix*Marseille université
Initiative d'excellence

Le cours de mécanique

Propriétés :

- **titre :** "Mécanique"
- **capacité :** 40 étudiants
- **étudiants inscrits :** initialement aucun

Comportements :

- **inscrire l'étudiant** `student` **:** si le nombre d'étudiants inscrits est égal à la capacité, rejeter la demande d'inscription. Sinon ajouter `student` aux étudiants inscrits.
- **fournir la liste des étudiants :** retourner la liste des étudiants.

Aix*Marseille université
Initiative d'excellence

Le cours d'informatique

Propriétés :

- **titre** : "Informatique"
- **capacité** : 60 étudiants
- **étudiants inscrits** : initialement aucun

Comportements :

- **inscrire l'étudiant** `student` : si le nombre d'étudiants inscrits est inférieur à la capacité, ajouter `student` aux étudiants inscrits. Sinon, inscrire `student` et désinscrire un étudiant choisi par tirage au sort.
- **fournir la liste des étudiants** : retourner la liste des étudiants.

Aix*Marseille université
Initiative d'excellence

Les classes

- Alice et Bob ont les mêmes comportements et les mêmes propriétés, seules les valeurs de leurs propriétés sont différentes. Alice et Bob appartiennent à la même classe.
- Mécanique et Informatique ne se comportent pas de la même façon, ils ne sont donc pas dans la même classe. Cependant ils proposent la même liste de services, ils ont donc une interface commune.

Aix✱Marseille université Initiative d'excellence

La classe des étudiants

Propriétés :

- **nom** : une chaîne de caractères.
- **préférences** : une liste de cours.

Comportements :

- **s'inscrire** : s'inscrit au premier cours de sa liste de préférences duquel `this` n'a pas encore été rejeté, jusqu'à ce que son inscription soit acceptée (ou avoir épuisé tous les cours).
- **se faire rejeter d'un cours** : s'inscrire.

Construction : nécessite de fournir un nom et une liste de préférence.

Aix*Marseille université
Initiative d'excellence

Qu'est-ce qu'une classe ?

Une classe

Une classe (d'objets) définit :

- la structure de ces objets (propriétés),
- le comportement de ces objets (méthodes).
- une façon d'initialiser ces objets (constructeur),

En Java, un programme est constitué de fichiers, chacun définissant une classe.

Aix*Marseille université Initiative d'excellence

Exécution d'un programme

Lors de l'exécution d'un programme, on constate :

- la création d'objets,
- l'envoi de message entre objets,
- le traitement des messages par une séquence d'instructions (qui peuvent être des créations d'objets ou des envois de messages, entre autres).

On peut représenter par des dessins :

- les objets par des rectangles, contenant les informations propres à l'objet (son *état*),
- les messages en cours de traitement par des flèches entre les objets.

Aix*Marseille université
Initiative d'excellence

Instructions du programme

```
Course info = new Course("info",2);
Course math = new Course("math",2);
Controller controller = new Controller();
Student alice = new Student("Alice", List.of(info,math));
controller.addStudent(alice);
Student bob = new Student("Bob", List.of(info,math));
controller.addStudent(bob);
Student carol = new Student("Carol", List.of(info,math));
controller.addStudent(carol);
controller.startRegistration();
```

Aix*Marseille université
Initiative d'excellence

Illustration

info : Course
name = "info" capacity = 2 enrolled = [] candidateCount = 0

Aix*Marseille université
Initiative d'excellence

Instructions du programme

```
Course info = new Course("info",2);
Course math = new Course("math",2);
Controller controller = new Controller();
Student alice = new Student("Alice", List.of(info,math));
controller.addStudent(alice);
Student bob = new Student("Bob", List.of(info,math));
controller.addStudent(bob);
Student carol = new Student("Carol", List.of(info,math));
controller.addStudent(carol);
controller.startRegistration();
```

Aix*Marseille université Initiative d'excellence

Illustration

info : Course
name = "info" capacity = 2 enrolled = [] candidateCount = 0

math : Course
name = "math" capacity = 2 enrolled = []

Aix*Marseille université
Initiative d'excellence

Instructions du programme

```
Course info = new Course("info",2);
Course math = new Course("math",2);
Controller controller = new Controller();
Student alice = new Student("Alice", List.of(info,math));
controller.addStudent(alice);
Student bob = new Student("Bob", List.of(info,math));
controller.addStudent(bob);
Student carol = new Student("Carol", List.of(info,math));
controller.addStudent(carol);
controller.startRegistration();
```


Aix*Marseille université
Initiative d'excellence

Illustration

info : Course
name = "info" capacity = 2 enrolled = [] candidateCount = 0

math : Course
name = "math" capacity = 2 enrolled = []

controller : Controller
students = []

Aix*Marseille université
Initiative d'excellence

Instructions du programme

```java
Course info = new Course("info",2);
Course math = new Course("math",2);
Controller controller = new Controller();
Student alice = new Student("Alice", List.of(info,math));
controller.addStudent(alice);
Student bob = new Student("Bob", List.of(info,math));
controller.addStudent(bob);
Student carol = new Student("Carol", List.of(info,math));
controller.addStudent(carol);
controller.startRegistration();
```

Aix*Marseille université
Initiative d'excellence

Illustration

info : Course
name = "info" capacity = 2 enrolled = [] candidateCount = 0

math : Course
name = "math" capacity = 2 enrolled = []

alice : Student
name = "Alice" pref = [info ; math] course =

controller : Controller
students = [Alice]

Aix*Marseille université Initiative d'excellence

Instructions du programme

```java
Course info = new Course("info",2);
Course math = new Course("math",2);
Controller controller = new Controller();
Student alice = new Student("Alice", List.of(info,math));
controller.addStudent(alice);
Student bob = new Student("Bob", List.of(info,math));
controller.addStudent(bob);
Student carol = new Student("Carol", List.of(info,math));
controller.addStudent(carol);
controller.startRegistration();
```

Aix*Marseille université
Initiative d'excellence

Illustration

info : Course

name = "info"
capacity = 2
enrolled = []
candidateCount = 0

math : Course

name = "math"
capacity = 2
enrolled = []

alice : Student

name = "Alice"
pref = [info ; math]
course =

bob : Student

name = "Bob"
pref = [info ; math]
course =

controller : Controller

students =
[Alice, Bob]

Aix*Marseille université
Initiative d'excellence

Instructions du programme

```java
Course info = new Course("info",2);
Course math = new Course("math",2);
Controller controller = new Controller();
Student alice = new Student("Alice", List.of(info,math));
controller.addStudent(alice);
Student bob = new Student("Bob", List.of(info,math));
controller.addStudent(bob);
Student carol = new Student("Carol", List.of(info,math));
controller.addStudent(carol);
controller.startRegistration();
```

Aix*Marseille université
Initiative d'excellence

Illustration

info : Course
name = "info" capacity = 2 enrolled = [] candidateCount = 0

math : Course
name = "math" capacity = 2 enrolled = []

alice : Student
name = "Alice" pref = [info ; math] course =

bob : Student
name = "Bob" pref = [info ; math] course =

carol : Student
name = "Carol" pref = [info ; math] course =

controller : Controller
students = [Alice, Bob, Carol]

Aix*Marseille université
Initiative d'excellence

Instructions du programme

```
Course info = new Course("info",2);
Course math = new Course("math",2);
Controller controller = new Controller();
Student alice = new Student("Alice", List.of(info,math));
controller.addStudent(alice);
Student bob = new Student("Bob", List.of(info,math));
controller.addStudent(bob);
Student carol = new Student("Carol", List.of(info,math));
controller.addStudent(carol);
controller.startRegistration();
```

Aix*Marseille université
Initiative d'excellence

Instructions du contrôleur

```
public void startRegistration() {
  for (Student s : students) {
    s.register();
  }
}
```

Aix*Marseille université
Initiative d'excellence

Instructions du contrôleur

```
public void startRegistration() {
  for (Student s : students) {
    s.register(); // s is Alice
  }
}
```

Aix*Marseille université
Initiative d'excellence

Illustration

info : Course
name = "info"
capacity = 2
enrolled = []
candidateCount = 0

math : Course
name = "math"
capacity = 2
enrolled = []

alice : Student
name = "Alice"
pref = [info ; math]
course =

bob : Student
name = "Bob"
pref = [info ; math]
course =

carol : Student
name = "Carol"
pref = [info ; math]
course =

controller : Controller
students =
[Alice, Bob, Carol]

Aix*Marseille université
Initiative d'excellence

Instructions des étudiants

```java
public void register() {
  Course course = preferences.pollFirst();
  boolean isAccepted = course.register(this);
  if (!isAccepted()) {
    this.register();
  }
}

public void reject() {
  this.register();
}
```

Aix*Marseille université
Initiative d'excellence

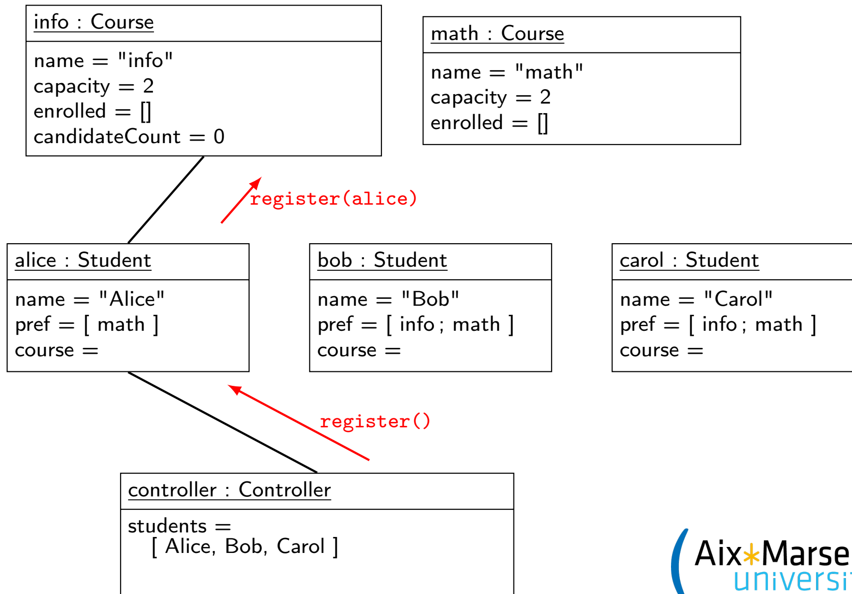
Instructions des étudiants

```
public void register() {
  Course course = preferences.pollFirst(); // info
  boolean isAccepted = course.register(this);
  if (!isAccepted()) {
    this.register();
  }
}

public void reject() {
  this.register();
}
```

Aix*Marseille université
Initiative d'excellence

Illustration



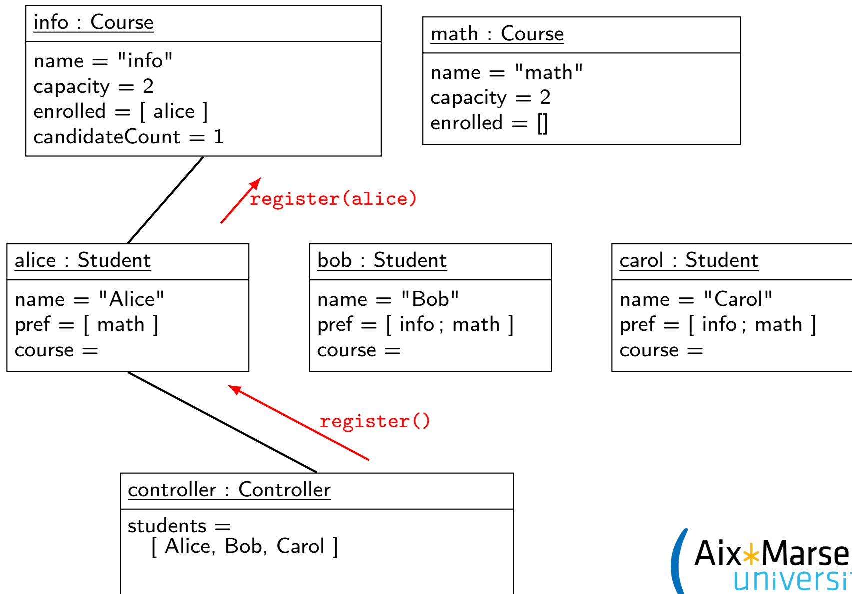
Aix*Marseille université
Initiative d'excellence

Instructions du cours d'info

```java
public void register(Student st) {
  registered.add(st);
  countRegistered = countRegistered + 1;
  if (registered.size() <= capacity) return true;
  int indexRemoved =
    Math.min(capacity,random.nextInt(countRegistered));
  Student rejected = registered.remove(indexRemoved);
  if (indexRemoved == capacity) return false;
  rejected.reject();
  return true;
}
```

Aix*Marseille université
Initiative d'excellence

Illustration



Aix*Marseille université
Initiative d'excellence

Instructions du cours d'info

```java
public void register(Student st) {
  registered.add(st);
  countRegistered = countRegistered + 1;
  if (registered.size() <= capacity) return true;
  int indexRemoved =
    Math.min(capacity,random.nextInt(countRegistered));
  Student rejected = registered.remove(indexRemoved);
  if (indexRemoved == capacity) return false;
  rejected.reject();
  return true;
}
```

Aix*Marseille université
Initiative d'excellence

Illustration

info : Course
name = "info"
capacity = 2
enrolled = [alice]
candidateCount = 1

math : Course
name = "math"
capacity = 2
enrolled = []

alice : Student
name = "Alice"
pref = [math]
course = info

bob : Student
name = "Bob"
pref = [info ; math]
course =

carol : Student
name = "Carol"
pref = [info ; math]
course =

controller : Controller
students =
[Alice, Bob, Carol]

Aix*Marseille université
Initiative d'excellence

Instructions des étudiants

```java
public void register() {
  Course course = preferences.pollFirst(); // info
  boolean isAccepted = course.register(this); // true
  if (!isAccepted()) {
    this.register();
  }
}

public void reject() {
  this.register();
}
```

Aix*Marseille université
Initiative d'excellence

Illustration

info : Course
name = "info"
capacity = 2
enrolled = [alice]
candidateCount = 1

math : Course
name = "math"
capacity = 2
enrolled = []

alice : Student
name = "Alice"
pref = [math]
course = info

bob : Student
name = "Bob"
pref = [info ; math]
course =

carol : Student
name = "Carol"
pref = [info ; math]
course =

controller : Controller
students =
[Alice, Bob, Carol]

Aix*Marseille université
Initiative d'excellence

Instructions du contrôleur

```java
public void startRegistration() {
  for (Student s : students) {
    s.register();
  }
}
```

Aix*Marseille université
Initiative d'excellence

Instructions du contrôleur

```
public void startRegistration() {
  for (Student s : students) {
    s.register(); // s is Bob
  }
}
```

Aix*Marseille université
Initiative d'excellence

Illustration

info : Course
name = "info"
capacity = 2
enrolled = [alice]
candidateCount = 1

math : Course
name = "math"
capacity = 2
enrolled = []

alice : Student
name = "Alice"
pref = [math]
course = info

bob : Student
name = "Bob"
pref = [info ; math]
course =

carol : Student
name = "Carol"
pref = [info ; math]
course =

controller : Controller
students =
[Alice, Bob, Carol]

Instructions des étudiants

```
public void register() {
  Course course = preferences.pollFirst();
  boolean isAccepted = course.register(this);
  if (!isAccepted()) {
    this.register();
  }
}

public void reject() {
  this.register();
}
```

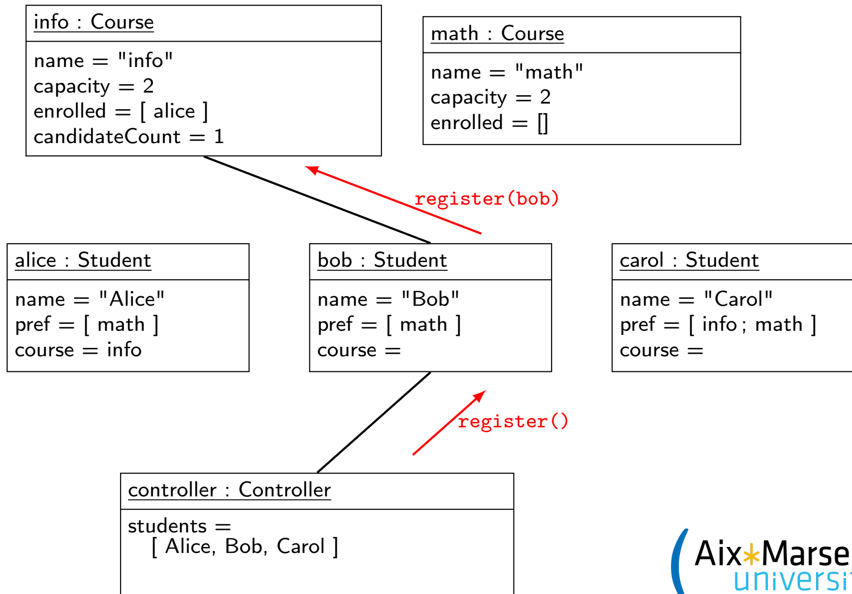
Aix*Marseille université
Initiative d'excellence

```
public void register() {
  Course course = preferences.pollFirst(); // info
  boolean isAccepted = course.register(this);
  if (!isAccepted()) {
    this.register();
  }
}

public void reject() {
  this.register();
}
```

Aix*Marseille université Initiative d'excellence

Illustration



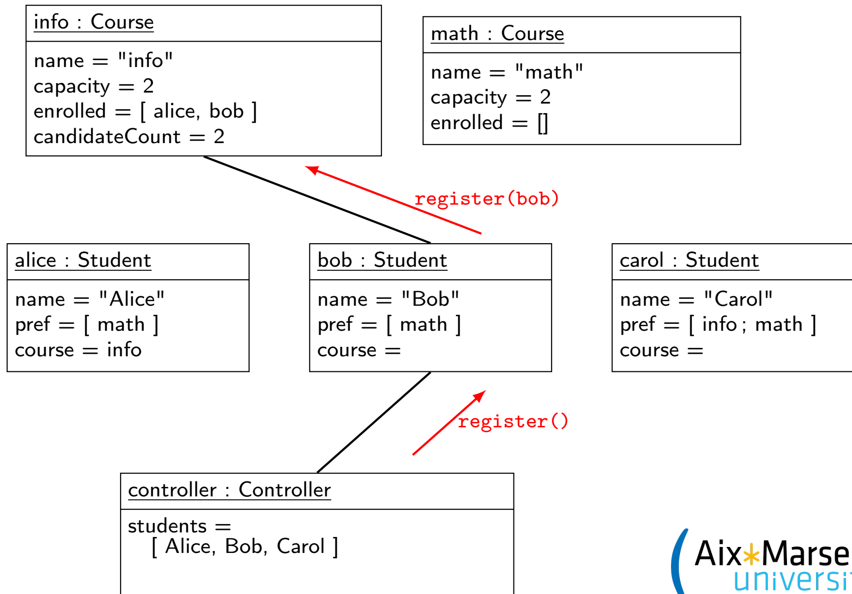
Aix*Marseille université
Initiative d'excellence

Instructions du cours d'info

```
public void register(Student st) {
  registered.add(st);
  countRegistered = countRegistered + 1;
  if (registered.size() <= capacity) return true;
  int indexRemoved =
    Math.min(capacity,random.nextInt(countRegistered));
  Student rejected = registered.remove(indexRemoved);
  if (indexRemoved == capacity) return false;
  rejected.reject();
  return true;
}
```

Aix*Marseille université
Initiative d'excellence

Illustration



Instructions du cours d'info

```java
public void register(Student st) {
  registered.add(st);
  countRegistered = countRegistered + 1;
  if (registered.size() <= capacity) return true;
  int indexRemoved =
    Math.min(capacity,random.nextInt(countRegistered));
  Student rejected = registered.remove(indexRemoved);
  if (indexRemoved == capacity) return false;
  rejected.reject();
  return true;
}
```

Aix*Marseille université
Initiative d'excellence

Illustration

info : Course

name = "info"
capacity = 2
enrolled = [alice, bob]
candidateCount = 2

math : Course

name = "math"
capacity = 2
enrolled = []

alice : Student

name = "Alice"
pref = [math]
course = info

bob : Student

name = "Bob"
pref = [math]
course = info

carol : Student

name = "Carol"
pref = [info ; math]
course =

register()

controller : Controller

students =
[Alice, Bob, Carol]

Aix*Marseille université
Initiative d'excellence

Instructions des étudiants

```java
public void register() {
  Course course = preferences.pollFirst(); // info
  boolean isAccepted = course.register(this); // true
  if (!isAccepted()) {
    this.register();
  }
}

public void reject() {
  this.register();
}
```

Aix*Marseille université
Initiative d'excellence

Illustration

info : Course

name = "info"
capacity = 2
enrolled = [alice, bob]
candidateCount = 2

math : Course

name = "math"
capacity = 2
enrolled = []

alice : Student

name = "Alice"
pref = [math]
course = info

bob : Student

name = "Bob"
pref = [math]
course = info

carol : Student

name = "Carol"
pref = [info ; math]
course =

controller : Controller

students =
[Alice, Bob, Carol]

Aix*Marseille université
Initiative d'excellence

Instructions du contrôleur

```
public void startRegistration() {
  for (Student s : students) {
    s.register();
  }
}
```

Aix*Marseille université
Initiative d'excellence

Instructions du contrôleur

```java
public void startRegistration() {
  for (Student s : students) {
    s.register(); // s is Carol
  }
}
```

Aix*Marseille université Initiative d'excellence

Illustration

info : Course

name = "info"
capacity = 2
enrolled = [alice, bob]
candidateCount = 2

math : Course

name = "math"
capacity = 2
enrolled = []

alice : Student

name = "Alice"
pref = [math]
course = info

bob : Student

name = "Bob"
pref = [math]
course = info

carol : Student

name = "Carol"
pref = [info ; math]
course =

register()

controller : Controller

students =
[Alice, Bob, Carol]

Aix*Marseille université
Initiative d'excellence

Instructions des étudiants

```java
public void register() {
  Course course = preferences.pollFirst();
  boolean isAccepted = course.register(this);
  if (!isAccepted()) {
    this.register();
  }
}

public void reject() {
  this.register();
}
```

Aix*Marseille université
Initiative d'excellence

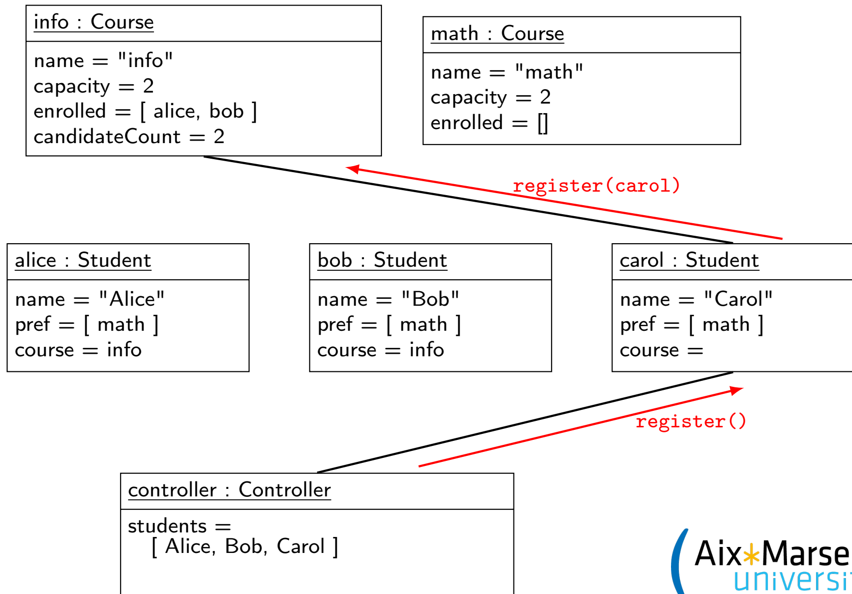
Instructions des étudiants

```java
public void register() {
  Course course = preferences.pollFirst(); // info
  boolean isAccepted = course.register(this);
  if (!isAccepted()) {
    this.register();
  }
}

public void reject() {
  this.register();
}
```

Aix*Marseille université Initiative d'excellence

Illustration



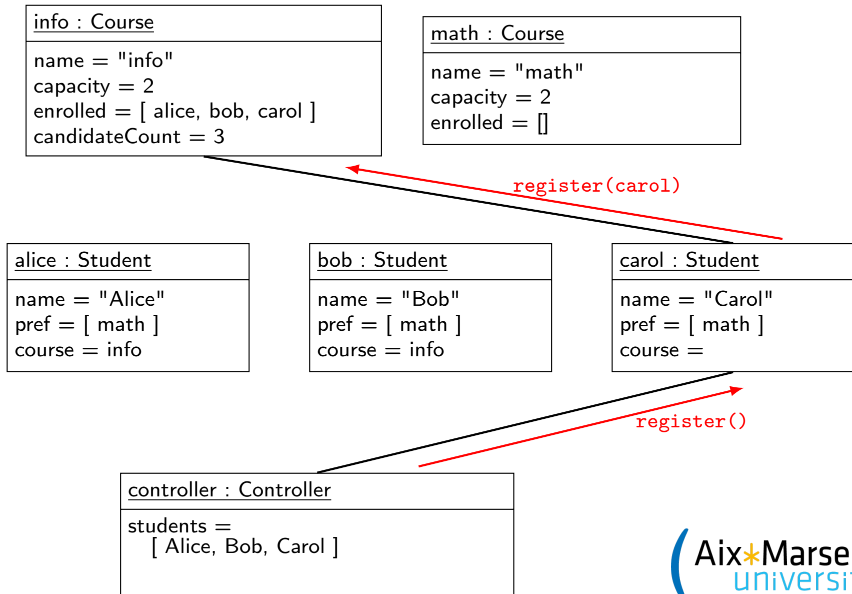
Aix*Marseille université
Initiative d'excellence

Instructions du cours d'info

```
public void register(Student st) {
  registered.add(st);
  countRegistered = countRegistered + 1;
  if (registered.size() <= capacity) return true;
  int indexRemoved =
    Math.min(capacity,random.nextInt(countRegistered));
  Student rejected = registered.remove(indexRemoved);
  if (indexRemoved == capacity) return false;
  rejected.reject();
  return true;
}
```

Aix*Marseille université
Initiative d'excellence

Illustration



Aix*Marseille université
Initiative d'excellence

Instructions du cours d'info

```
public void register(Student st) {
  registered.add(st);
  countRegistered = countRegistered + 1;
  if (registered.size() <= capacity) return true;
  int indexRemoved =
    Math.min(capacity,random.nextInt(countRegistered));
  Student rejected = registered.remove(indexRemoved);
  if (indexRemoved == capacity) return false;
  rejected.reject();
  return true;
}
```

Aix*Marseille université
Initiative d'excellence

```
public void register(Student st) {
  registered.add(st);
  countRegistered = countRegistered + 1;
  if (registered.size() <= capacity) return true;
  int indexRemoved =
    Math.min(capacity,random.nextInt(countRegistered));
  Student rejected = registered.remove(indexRemoved);
  if (indexRemoved == capacity) return false;
  rejected.reject();
  return true;
}
```

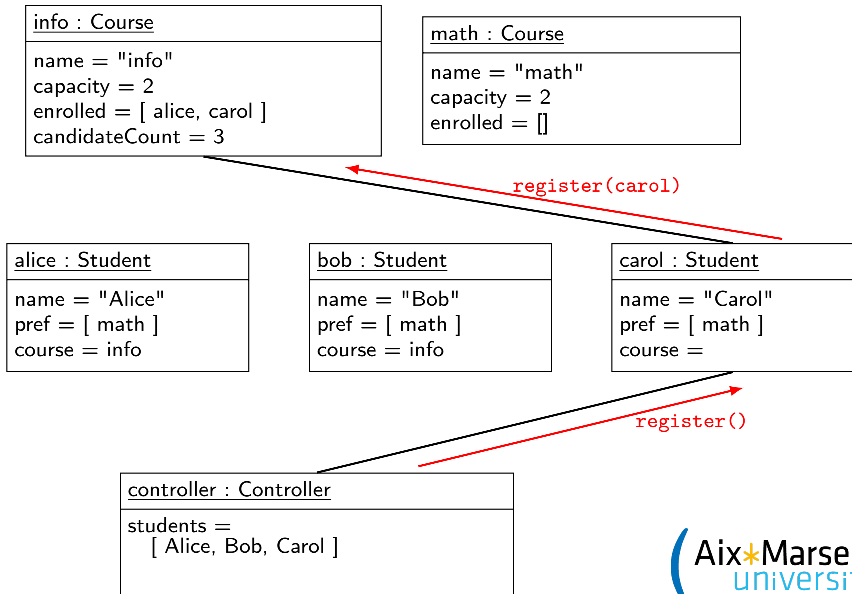
Aix*Marseille université
Initiative d'excellence

Instructions du cours d'info

```
public void register(Student st) {
  registered.add(st);
  countRegistered = countRegistered + 1;
  if (registered.size() <= capacity) return true;
  int indexRemoved = // 2
    Math.min(capacity,random.nextInt(countRegistered));
  Student rejected = registered.remove(indexRemoved);
  if (indexRemoved == capacity) return false;
  rejected.reject();
  return true;
}
```

Aix*Marseille université
Initiative d'excellence

Illustration



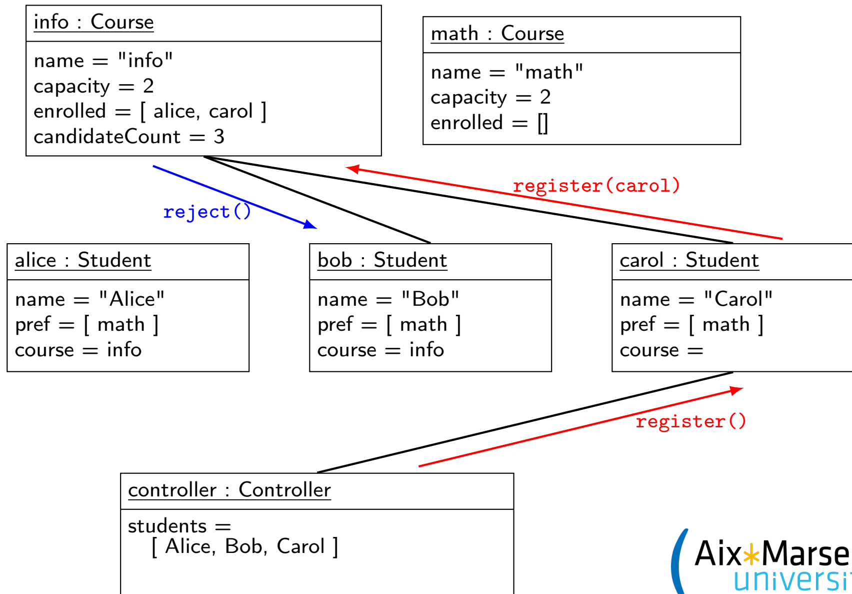
Aix*Marseille université
Initiative d'excellence

Instructions du cours d'info

```
public void register(Student st) {
  registered.add(st);
  countRegistered = countRegistered + 1;
  if (registered.size() <= capacity) return true;
  int indexRemoved = // 2
    Math.min(capacity,random.nextInt(countRegistered));
  Student rejected = registered.remove(indexRemoved); // bob
  if (indexRemoved == capacity) return false;
  rejected.reject();
  return true;
}
```

Aix*Marseille université
Initiative d'excellence

Illustration



Aix*Marseille université
Initiative d'excellence

Instructions des étudiants

```java
public void register() {
  Course course = preferences.pollFirst();
  boolean isAccepted = course.register(this);
  if (!isAccepted()) {
    this.register();
  }
}

public void reject() {
  this.register();
}
```

Aix*Marseille université
Initiative d'excellence

Instructions des étudiants

```
public void register() {
  Course course = preferences.pollFirst();
  boolean isAccepted = course.register(this);
  if (!isAccepted()) {
    this.register();
  }
}

public void reject() {
  this.register();
}
```

Aix*Marseille université
Initiative d'excellence

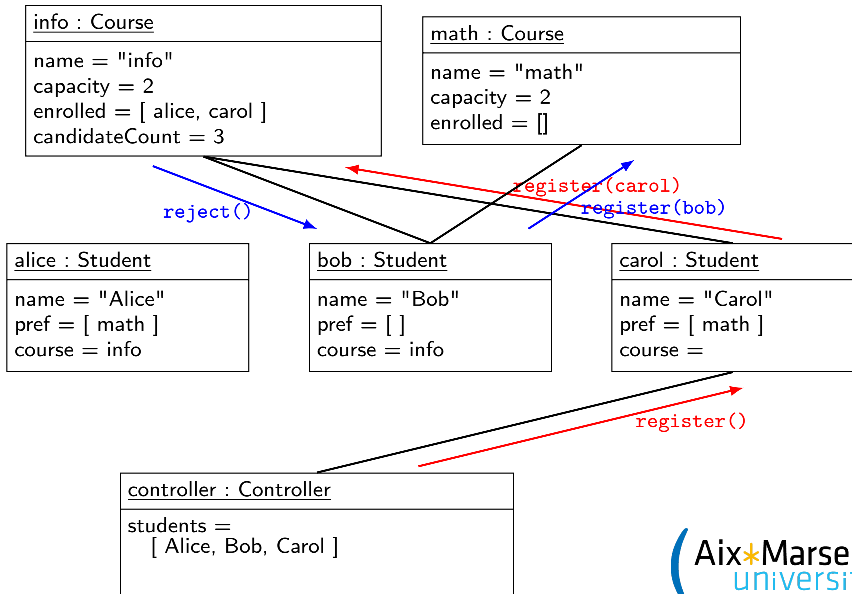
Instructions des étudiants

```
public void register() {
  Course course = preferences.pollFirst(); // maths
  boolean isAccepted = course.register(this);
  if (!isAccepted()) {
    this.register();
  }
}

public void reject() {
  this.register();
}
```

Aix*Marseille université Initiative d'excellence

Illustration



Aix*Marseille université
Initiative d'excellence

Instructions du cours de maths

```
public void register(Student st) {
  if (registered.size() >= capacity) return false;
  registered.add(st);
  return true;
}
```

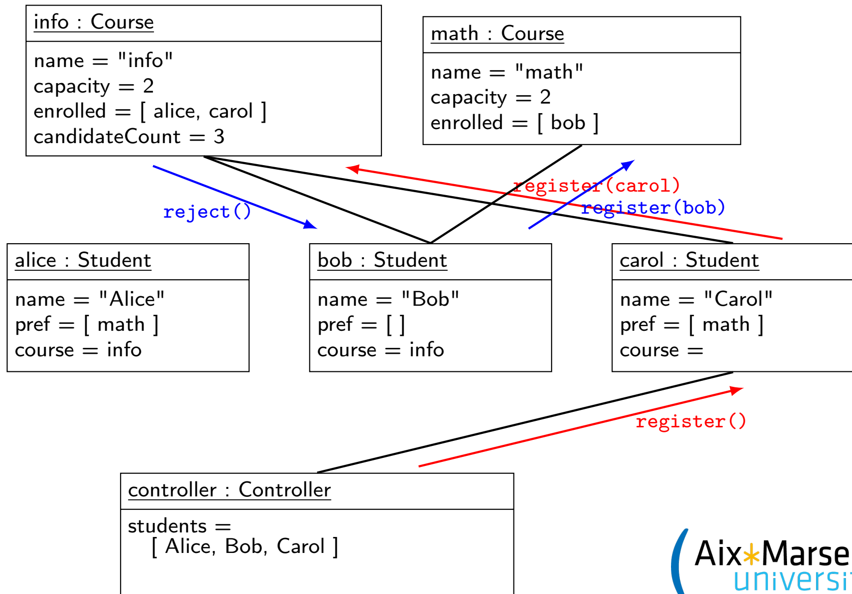
Aix*Marseille université
Initiative d'excellence

Instructions du cours de maths

```java
public void register(Student st) {
  if (registered.size() >= capacity) return false;
  registered.add(st);
  return true;
}
```

Aix*Marseille université
Initiative d'excellence

Illustration

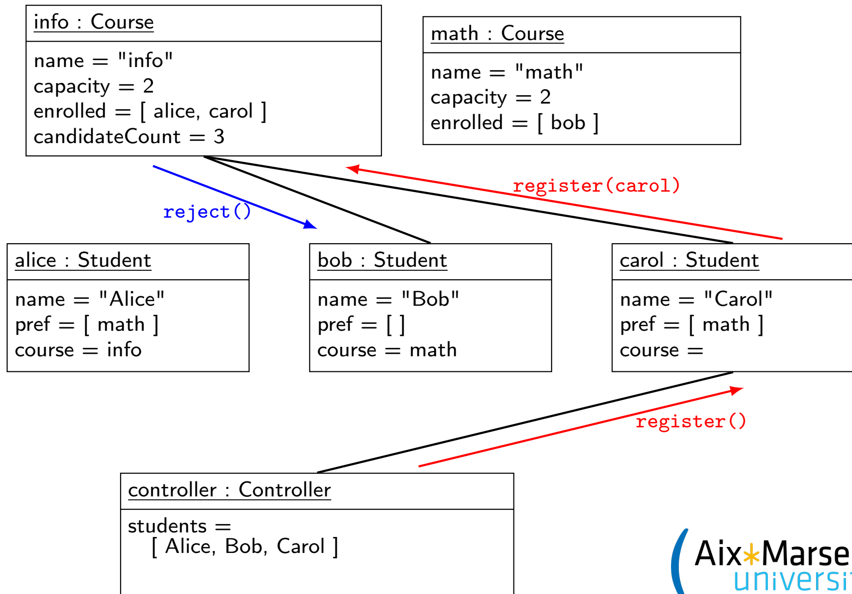


Instructions du cours de maths

```
public void register(Student st) {
  if (registered.size() >= capacity) return false;
  registered.add(st);
  return true;
}
```

Aix*Marseille université
Initiative d'excellence

Illustration



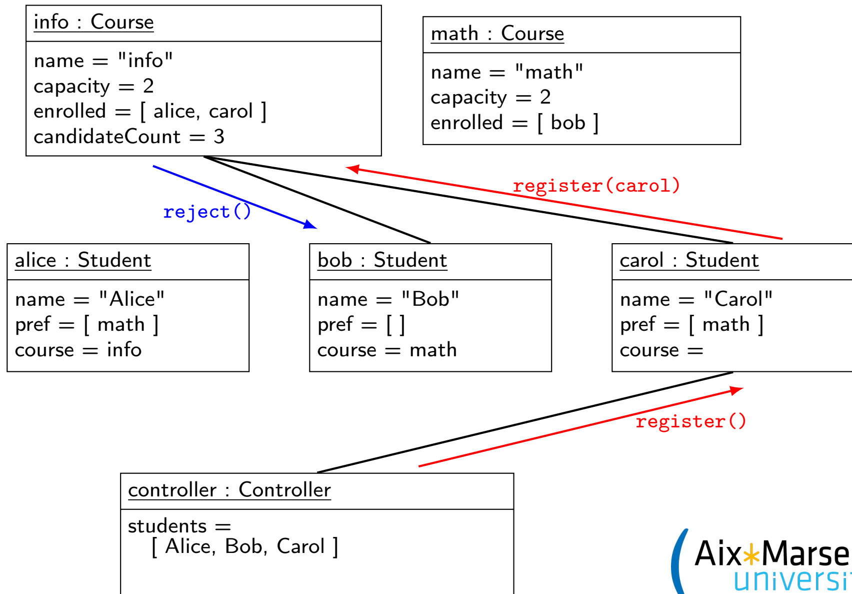
Instructions des étudiants

```java
public void register() {
  Course course = preferences.pollFirst(); // maths
  boolean isAccepted = course.register(this);  // true
  if (!isAccepted()) {
    this.register();
  }
}

public void reject() {
  this.register();
}
```

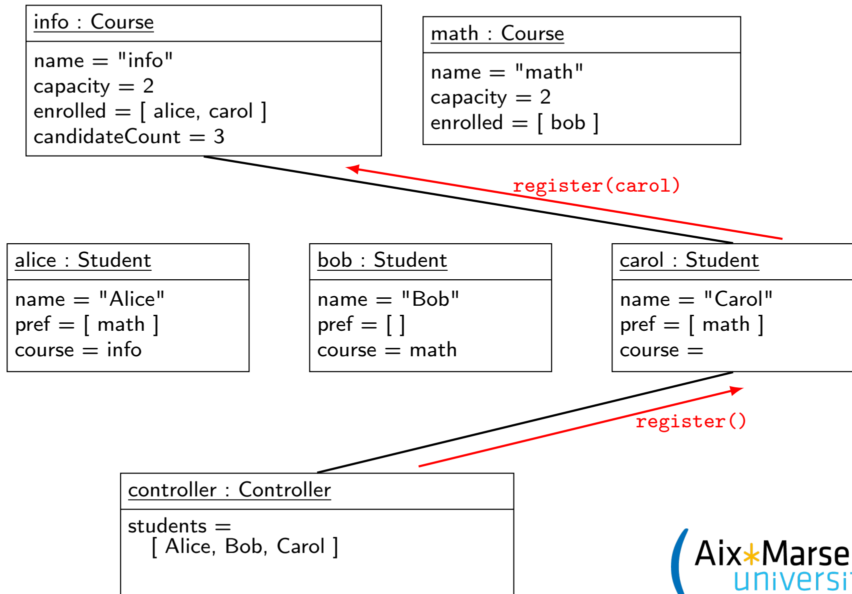
Aix*Marseille université
Initiative d'excellence

Illustration



Aix*Marseille université Initiative d'excellence

Illustration



Aix*Marseille université
Initiative d'excellence

```java
public void register(Student st) {
  registered.add(st);
  countRegistered = countRegistered + 1;
  if (registered.size() <= capacity) return true;
  int indexRemoved = // 2
    Math.min(capacity,random.nextInt(countRegistered));
  Student rejected = registered.remove(indexRemoved); // bob
  if (indexRemoved == capacity) return false;
  rejected.reject();
  return true;
}
```

Illustration

info : Course
name = "info"
capacity = 2
enrolled = [alice, carol]
candidateCount = 3

math : Course
name = "math"
capacity = 2
enrolled = [bob]

alice : Student
name = "Alice"
pref = [math]
course = info

bob : Student
name = "Bob"
pref = [ ]
course = math

carol : Student
name = "Carol"
pref = [math]
course = info

controller : Controller
students =
[Alice, Bob, Carol]

Aix*Marseille université
Initiative d'excellence

Instructions des étudiants

```
public void register() {
  Course course = preferences.pollFirst(); // info
  boolean isAccepted = course.register(this); // true
  if (!isAccepted()) {
    this.register();
  }
}

public void reject() {
  this.register();
}
```

Aix*Marseille université
Initiative d'excellence

Illustration

info : Course

name = "info"
capacity = 2
enrolled = [alice, carol]
candidateCount = 3

math : Course

name = "math"
capacity = 2
enrolled = [bob]

alice : Student

name = "Alice"
pref = [math]
course = info

bob : Student

name = "Bob"
pref = [ ]
course = math

carol : Student

name = "Carol"
pref = [math]
course = info

controller : Controller

students =
[Alice, Bob, Carol]

Aix*Marseille université
Initiative d'excellence

La classe `Student` : déclaration

```
public class Student {

  ... /* see next slides */

}
```

- `class` introduit une classe,
- `Student` est le nom donné à cette classe,
- `public` précise que cette classe est connue de tous les objets.
- les `{ }` délimitent le contenu de la classe.

Aix*Marseille université Initiative d'excellence

La classe Student : propriétés

```
public final String name;
private final List<Course> prefs;
private Course course;
```

- name, prefs, course sont les trois propriétés des objets de la classe Student
- String, List<Course> et Course sont les classes des objets de chaque propriétés.
- public ou private indique si la propriété est connue de tous les objets ou seulement ceux de la classe.
- final indique que la propriété ne change jamais de valeur après l'initialisation de l'objet.

Aix*Marseille université Initiative d'excellence

La classe `Student` : constructeur

```
public Student(String studentName,
               List<Course> preferredCourses) {
  this.name = studentName;
  this.prefs = preferredCourses;
}
```

- Entre parenthèses, les paramètres transmis lors de la création de l'objet.
- `this.name` est la propriété `name` de l'objet en construction `this`.
- `this.name = studentName` initialise la propriété `name` de l'objet en construction à la valeur `studentName` donnée au constructeur.

Aix*Marseille université Initiative d'excellence

La classe Student : isRegistered

```
public boolean isRegistered() {
  return currentCourse != null;
}
```

- `isRegistered` est une méthode, elle retourne une réponse *vrai* ou *faux* (`boolean`) à l'appelant,
- `public` indique que tout objet peut l'appeler,
- `return` permet de définir le message retourné.

Aix*Marseille université
Initiative d'excellence

La classe Student : register

```
public void register() {
  Course course = preferences.pollFirst();
  boolean isAccepted = course.register(this);
  if (!isAccepted()) {
    this.register();
  }
}
```

Aix*Marseille université Initiative d'excellence

La classe Student : reject

```
public void reject() {
  currentCourse = null;
  register();
}
```

- null est une absence de valeur,
- un objet peut appeler l'une de ses méthodes (ici register).

Aix*Marseille université Initiative d'excellence