Conception objet en Java avec BlueJ
une approche interactive

12. Conception des applications

David J. Barnes, Michael Kölling
version française: Patrice Moreaux

Principaux concepts abordés

- Découvrir les classes
- Cartes CRC
- Concevoir les interfaces
- Patrons de conception

Analyse et conception

- Un domaine vaste et complexe.
- La méthode nom/verbe est adaptée aux problèmes relativement petits.
- On utilise les cartes CRC dans la phase de conception.

La méthode nom/verbe

- Dans une description, les noms font référence aux « choses ».
  - sources des classes et des objets.
- Les verbes font référence aux actions.
  - sources des interactions entre objets.
  - les actions sont des comportements donc des méthodes.

Une description de problème

Le système de réservation doit enregistrer les réservations
de places de cinéma de plusieurs salles.

Chaque salle comporte des sièges organisés par rangée.
Les clients peuvent réserver des places; on leur attribue
un numéro de rangée et un numéro de place.
Ils peuvent demander plusieurs places voisines.

Une réservation concerne une séance particulière (un film à une heure donnée).
Les séances ont lieu à une date et une heure données dans une salle donnée.

Le système enregistre le numéro de téléphone du client.

Noms et verbes

Système de réservation
Stocke (réservation places)
Stocke (numéro de téléphone)

Salle
a (sièges)

Film

Client
Réserve (places)
Reçoit (numéro de rangée, de place)
Demande (réservation places)

Heure

Date

Réservation places

Séance
a lieu (dans salle)

Siège

Numéro siège

Numéro Téléphone

Rangée

Numéro rangée

Utiliser les cartes CRC

- Introduites par Kent Beck et Ward Cunningham.
- Chaque carte comporte:
  - un nom de *classe (C).*
  - les *responsabilités* de la classe (R).
  - les *collaborateurs* de la classe (C).

Une carte CRC

Nom de la classe	**Collaborateurs**
Responsabilités	

Scénario

- Une activité que le système doit réaliser ou gérer.
  - quelque fois appelé *cas d'utilisation*.
- Utilisé pour découvrir et noter les interactions entre objets (collaborations).
- À réaliser (jouer) en groupe.

Un exemple partiel

CinemaBookingSystem	*Collaborateurs*
peut trouver les séances par titre et jour. Stocke des collection de séances. Extrait et affiche les informations d'une séance.	séance Collection

...

Scénarios comme moyen d'analyse

- Les scénarios servent à vérifier que la description du problème est claire et complète.
- Passez suffisamment de temps sur l'analyse.
- L'analyse conduit à la conception.
  - mettre en évidence des erreurs ou des oublis à ce moment fait gagner beaucoup de travail pour la suite.

Conception de classe

- L'analyse par scénario permet de clarifier la structure de l'application.
  - chaque carte fournit une classe.
  - les collaborations mettent en évidence les coopérations et interactions entre objets.
- Les responsabilités fournissent les méthodes publiques.
  - et parfois des champs; ex. « stocke une collection ... »

Conception des interfaces de classes

- Rejouez les scénarios en termes d'appels de méthodes, paramètres et valeurs de retour.
- Notez les signatures obtenues.
- Créez les schémas de classes avec les squelettes de méthodes publiques.
- Une conception minutieuse est la clé d'une implantation réussie.

Documentation

- Écrire les commentaires de classe.
- Écrire les commentaires de méthode.
- Décrire le but général de chacune.
- Documenter dès maintenant garantit que:
  - on se concentre sur *quoi* plutôt que *comment*.
  - on n'oubliera pas de commenter!

Coopération

- Le travail en équipe tend à devenir la norme et non l'exception.
- La documentation est essentielle pour le travail en équipe.
- Une conception OO « propre » avec des composants faiblement couplés favorise aussi la coopération.

Prototypage

- Pour analyser le système au plus tôt.
  - Identification précoce des problèmes.
- On peut simuler des composants incomplets.
  - par ex. en retournant un résultat constant.
  - éviter les comportements aléatoires, difficiles à reproduire!

Croissance du logiciel

- Modèle de la chute d'eau.
  - analyse
  - conception
  - implantation
  - test unitaire
  - test d'intégration
  - recette (fourniture au client)
- Pas d'itération prévue.

Développement itératif

- Utiliser un prototype au plus tôt.
- Interactions fréquentes avec le client.
- Boucle sur:
  - analyse
  - conception
  - prototype
  - retour client
- Un modèle avec accroissement prévu du projet est le plus réaliste.

Utiliser les patrons de conception

- Les relations inter-classes sont importantes et peuvent être complexes.
- On retrouve certaines relations dans de nombreuses applications.
- Les patrons de conception ("Design patterns") aident à clarifier les relations et encouragent la réutilisation.

Structure d'un patron

- Un nom de patron.
- Le problème associé.
- En quoi il fournit une solution:
  - structures, participants, collaborations.
- Résultats de son emploi.
  - résultats, compromis.

Décorateur

- Accroît les fonctionnalités d'un objet.
- Un objet décorateur enveloppe un autre objet.
  - Le décorateur a la même interface.
  - les appels sont transmis à l'objet enveloppé ...
  - ... mais le décorateur peut intercaler certaines actions.
- Exemple: **`java.io.BufferedReader`**
  - enveloppe et étend un objet **`Reader`** non tamponné.

Singleton

- Garantit qu'une seule instance d'une classe existe.
  - tous les clients utilisent le même objet.
- Le constructeur est privé pour empêcher des instanciations externes.
- On obtient une instance unique avec un appel à la méthode statique **`getInstance`**.
- Exemple: **`Canvas`** du projet *shapes*.

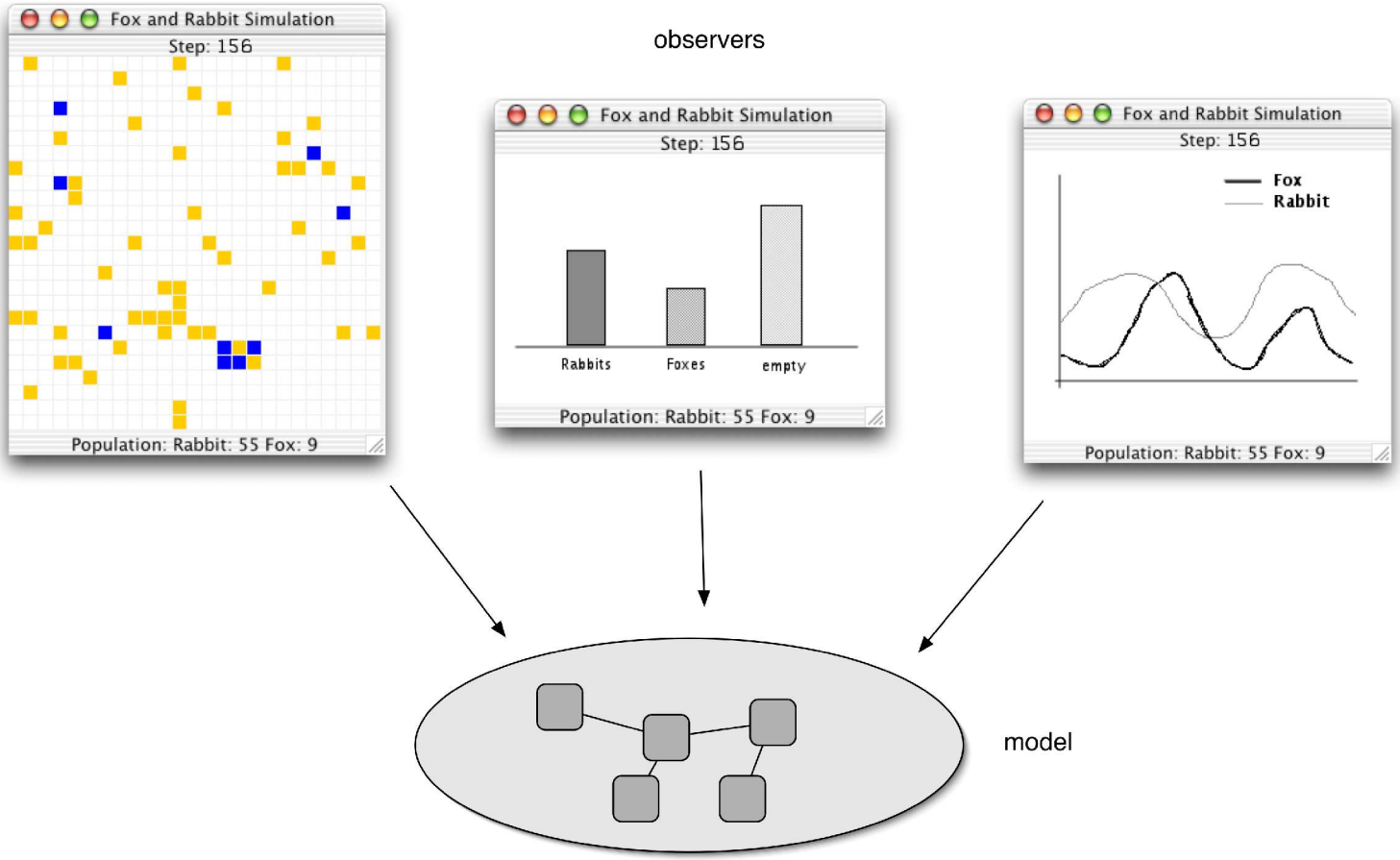
La méthode fabrique ("factory")

- Patron de création.
- Les clients demandent un objet d'un type interface ou d'une superclasse donnée.
- Une méthode fabrique un objet de la classe ou d'une sous-classe.
- Le type retourné dépend du contexte.
- Exemple: les méthodes **`iterator`** des classes Collection.

Observateur

- Pour séparer le modèle interne de la vue de ce modèle.
- Un observateur définit une relation un-à-plusieurs entre objets.
- L'objet observé informe tous ses observateurs de tout changement d'état.
- Exemple: **`SimulatorView`** du projet *foxes-and-rabbits*.

Observateurs



Résumé (1)

- Les collaborations entre classes et les interactions entre objets doivent être identifiés.
  - l'analyse CRC facilite cette identification.
- Une approche itérative pour concevoir, analyser et implanter peut être bénéfique.
  - percevoir les systèmes logiciels comme des entités qui croîtront et évolueront dans le temps.

Résumé (2)

- Travaillez de manière à faciliter les collaborations avec les autres acteurs.
- Concevez des structures de classe souples et extensibles.
  - aide: connaître les patrons de conception existant.
- Continuez à apprendre par votre propre travail et l'expérience des autres.

Sommaire général