Le modèle « State » (1/5)

- Nom
  - State (modèle comportemental de niveau objet)
- Intention
  - Permettre à un objet d'adapter son comportement en fonction de son état interne
- Motivation
  - L'approche classique qui consiste à utiliser des conditions dans le corps des méthodes conduit à des méthodes complexes
  - En réifiant l'état sous forme d'objet (1 classe par état possible) et en déléguant le traitement de la méthode à cet objet, on rend le traitement spécifique à l'état courant de la machine à états
- Indications d'utilisation
  - Quand le comportement d'un objet dépend de son état et que l'implantation de cette dépendance à l'état par des instructions conditionnelles est trop complexe

Le modèle « State » (2/5)

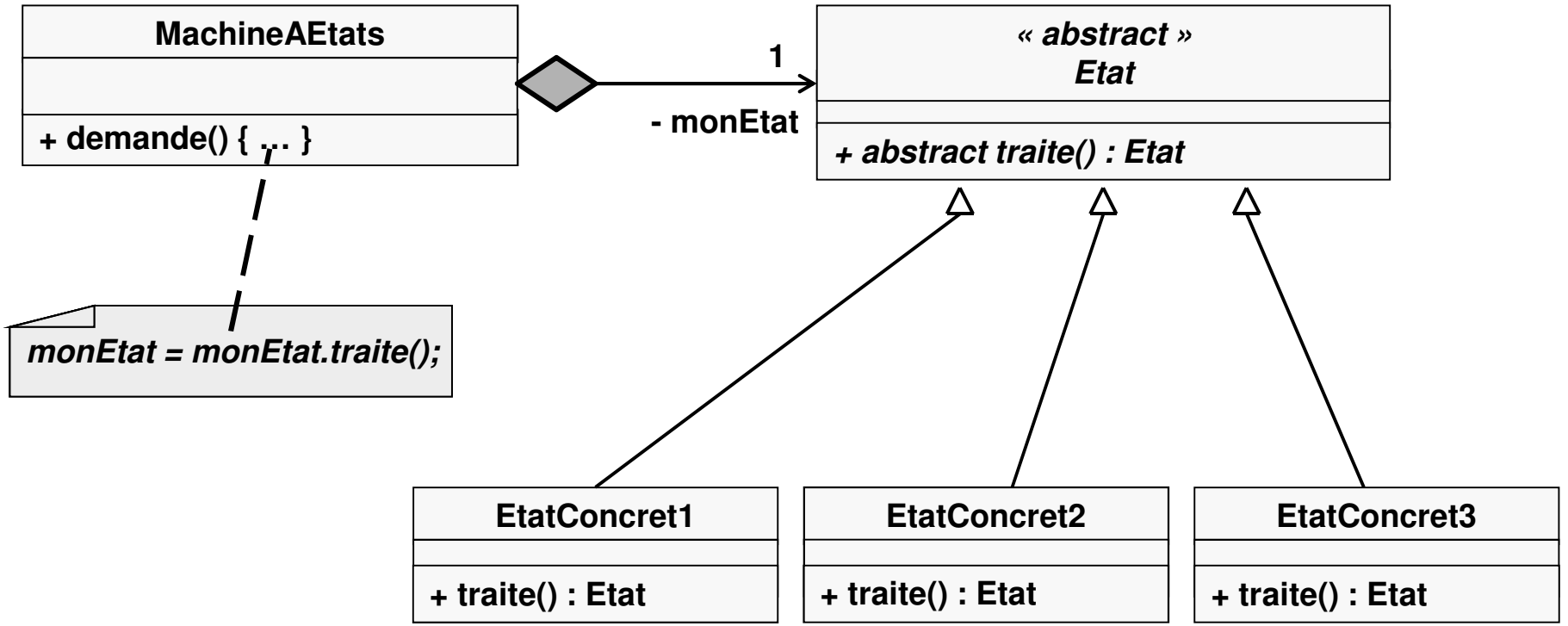
- Participants
  - *MachineAEtat* : classe concrète définissant des objets qui sont des machines à états (pouvant être décrits par un diagramme d'états-transitions). Cette classe maintient une référence vers une instance d'état qui définit l'état courant
  - *Etat* : classe abstraite qui spécifie les méthodes liées à l'état et qui gère l'association avec la machine à états
  - *Etatconcret1*... : (sous-)classes concrètes qui implantent le comportement de *MachineAEtat* pour chacun des ses états

Le modèle « State » (3/5)

- Collaborations
  - La *MachineAEtat* transmet la requête à l'objet *EtatConcret* qui la traite par « délégation » (sous-traitance) ; ce traitement provoque la mise à jour de l'état de MachineAEtat
    - C'est l'objet *EtatConcret* qui décide du nouvel état (EtatConcret) de *MachineAEtat* et initie la mise à jour
    - Le nouvel objet *EtatConcret* est retourné en résultat du traitement (cf. signatures des méthodes dans le diagramme de classes)
      - Alternativement (variante), *EtatConcret* peut utiliser une méthode de **callback** offerte par *MachineAEtat*

Le modèle « State » (4/5)

- Structure



Le modèle « State » (5/5)

- Modèle apparenté
  - Stratégie
    - Fondamentalement, State diffère de Stratégie par son intention. State a pour intention de permettre à un objet d'adapter son comportement en fonction de son état et de changer cet état
    - Dans la solution (mise en œuvre), le changement de stratégie est externe (méthode setStratégie()) alors que c'est l'état lui-même qui provoque le changement d'état (opération interne)