CS 351
Design of Large Programs
Abstract Data Types

Brooke Chenoweth

University of New Mexico

Spring 2024

Data Abstraction Revisited

- Built-in data types (int, boolean, etc.)
- Programmer-defined types
- Strongly-typed languages
- Abstract data type (ADT)
  - a formal characterization of a set of data structures
  - sharing a common set of operations
- Generic types (parameterized definitions)

Abstract Data Type Definition

A formal characterization of a set of data structures that share a common set of operations having well-defined syntax and semantics.

An ADT specification

- is independent of any possible realization
- may be captured in purely mathematical terms

The ADT is the conceptual basis for the class construct

Basic Class Concept

The notion of class assumes many forms:

- mathematics
  - collection of sets sharing some property
- natural language concept
  - collection of objects sharing some properties
  - red, car, birds, etc.
- design notation
  - documentation of a set of objects having identical properties
  - does not depend on availability of an object-oriented programming language
- programming language construct

Class Construct in Java

Embodiment of the abstract data type concept

- fields
- methods

Mechanisms for deriving new classes:

- inheritance
  - single (extending a class)
  - multiple (implementing interfaces)
- new fields and methods
- method overriding
- inheritance controls (final)

Access control mechanics:

- public, private, protected

Sample Class Definition

```
public class Asteroid {
  private static int nextid = 0;
  private int id;
  private Color color;
  private Point location;
  private int[] velocity;

  public Asteroid(Color color, Point location, int[] velocity) {
    this.color = color;
    this.location = location;
    this.velocity = velocity;
    this.id = nextid;
    nextid++;
  }

  public void updateLocation(int elapsedTime) {
    // ...
  }

  public void setVelocity(int[] velocity) {
    this.velocity = velocity;
  }

  public int[] getVelocity() { return velocity; }
  public Point getLocation() { return location; }
}
```

OOD Perspective

Class as a strict embodiment of the abstract data type concept

- private fields
- public methods

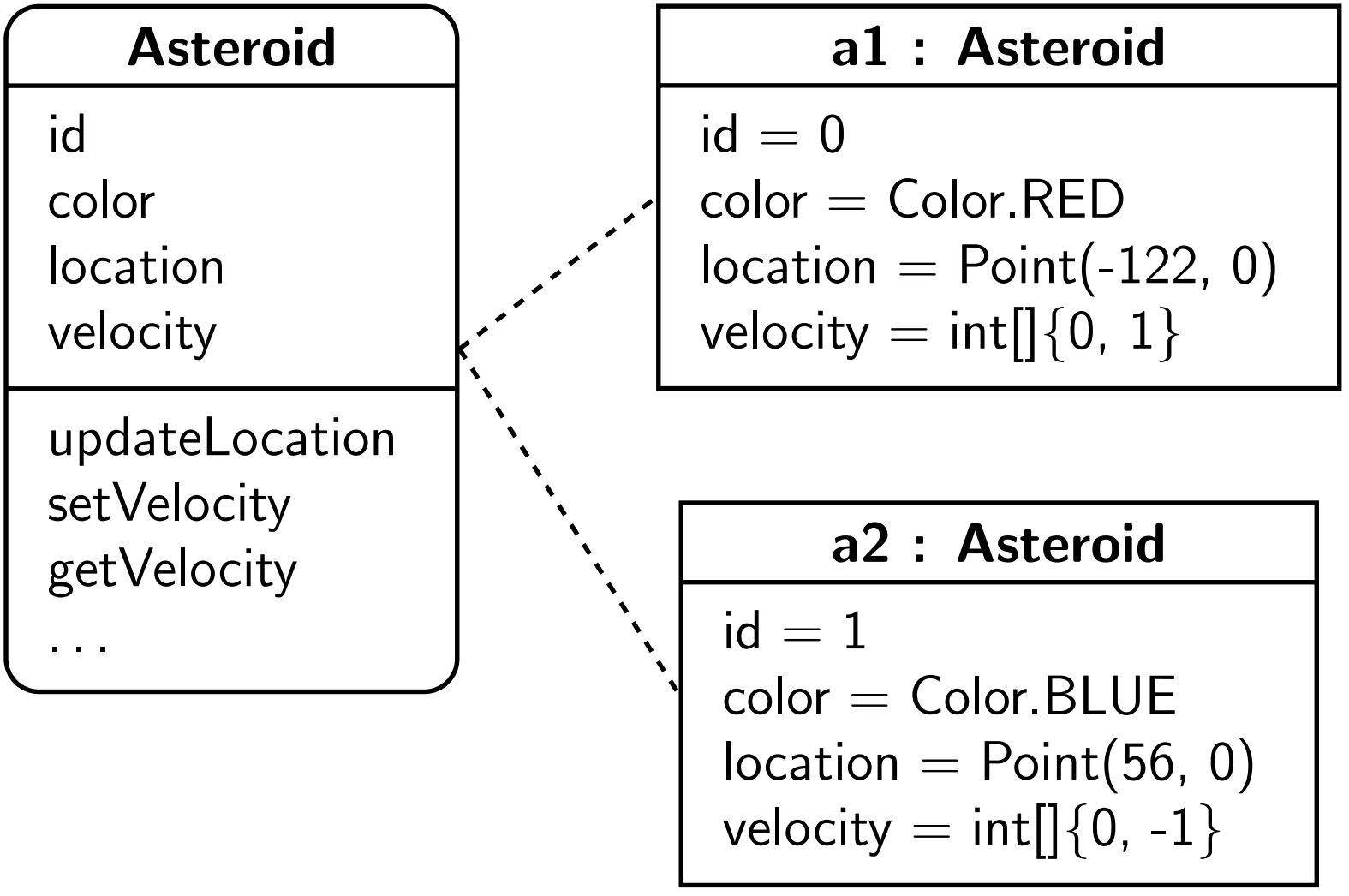
Restricted mechanisms for deriving new classes

- inheritance
  - single (extending a class)
  - multiple (implementing interfaces)
- method overriding subject to semantic consistency

Object Creation

- Objects are dynamically created instances of a class
- Storage is allocated for the fields
- Code is reused from the class definition
- Java uses garbage collection to reclaim storage used by inaccessible objects

Notation for Instantiation



Simple Design Diagram