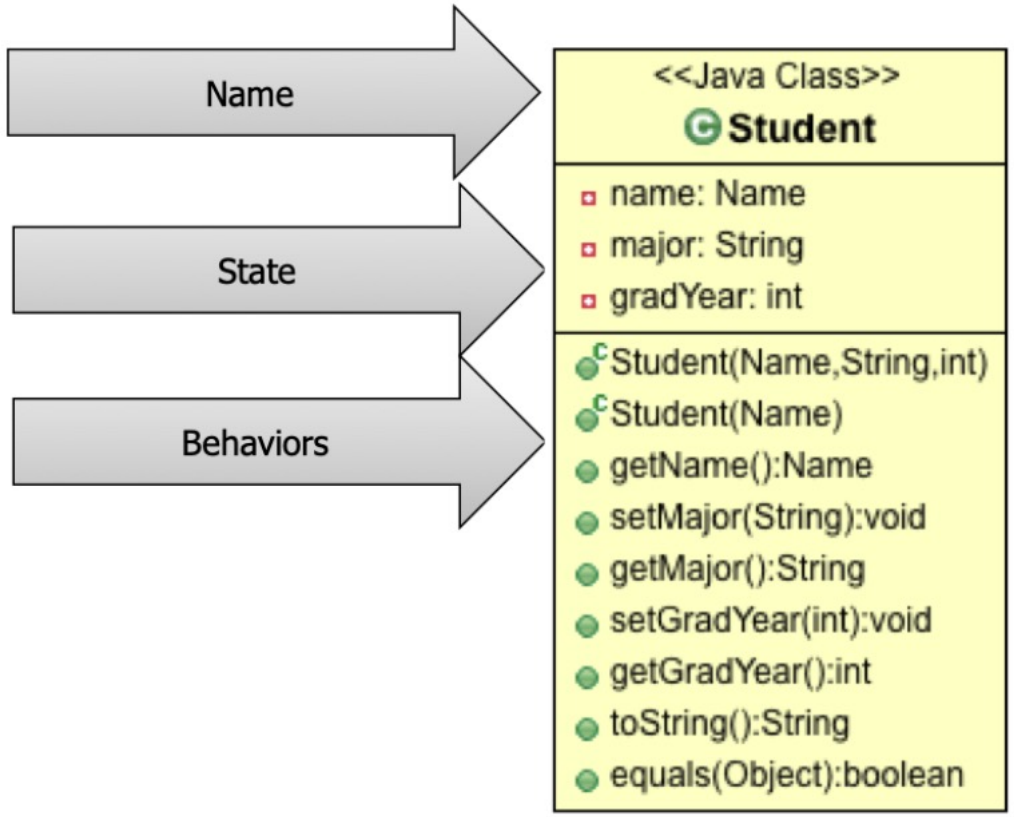
INTERACTING CLASSES AND OBJECT-ORIENTED DESIGN

zyBook 9.6, zyBook 9.9, zyBook 9.14

@ Dr. Jessica Young Schmidt and NCSU Computer Science Faculty

SOFTWARE DESIGN

- Goal
  - decide the structure of the software and the hardware configurations that support it
- How individual classes and software components work together in the software system
- Software Artifacts
  - design documents, class diagrams, other UML diagrams



UML Diagrams

- UML: Unified Modeling Language
  - Models object-oriented software
  - Convergence from three earlier modeling languages
    - OMT (James Rumbaugh)
    - OOSE (Ivar Jacobson)
    - Booch (Grady Booch)
- Overseen by the Object Mentor Group (OMG): (`www.omg.org`)
- Access modifiers
  - - or red square: private
  - + or green circle: public

CLASS DESIGN

- Do not provide any functionality that does not have a clear use
- Omit mutators that have no use
- Limit object creation to the constructor
- "Classes should be immutable unless there's a very good reason to make them mutable... If a class cannot be made immutable, limit it's mutability as much as possible." - Joshua Bloch
- Classes should have cohesion
  - The extent to which the code for a class represents a single abstraction
  - Allows for reusability of the class in other programs
  - For example
    - The Book class represents only things a book can do and knows
    - A Book should not perform console input and output
- Classes should not have unnecessary dependencies
  - Coupling is the degree to which one part of a program depends on another
  - Related data and behavior should be in the same place (same class)

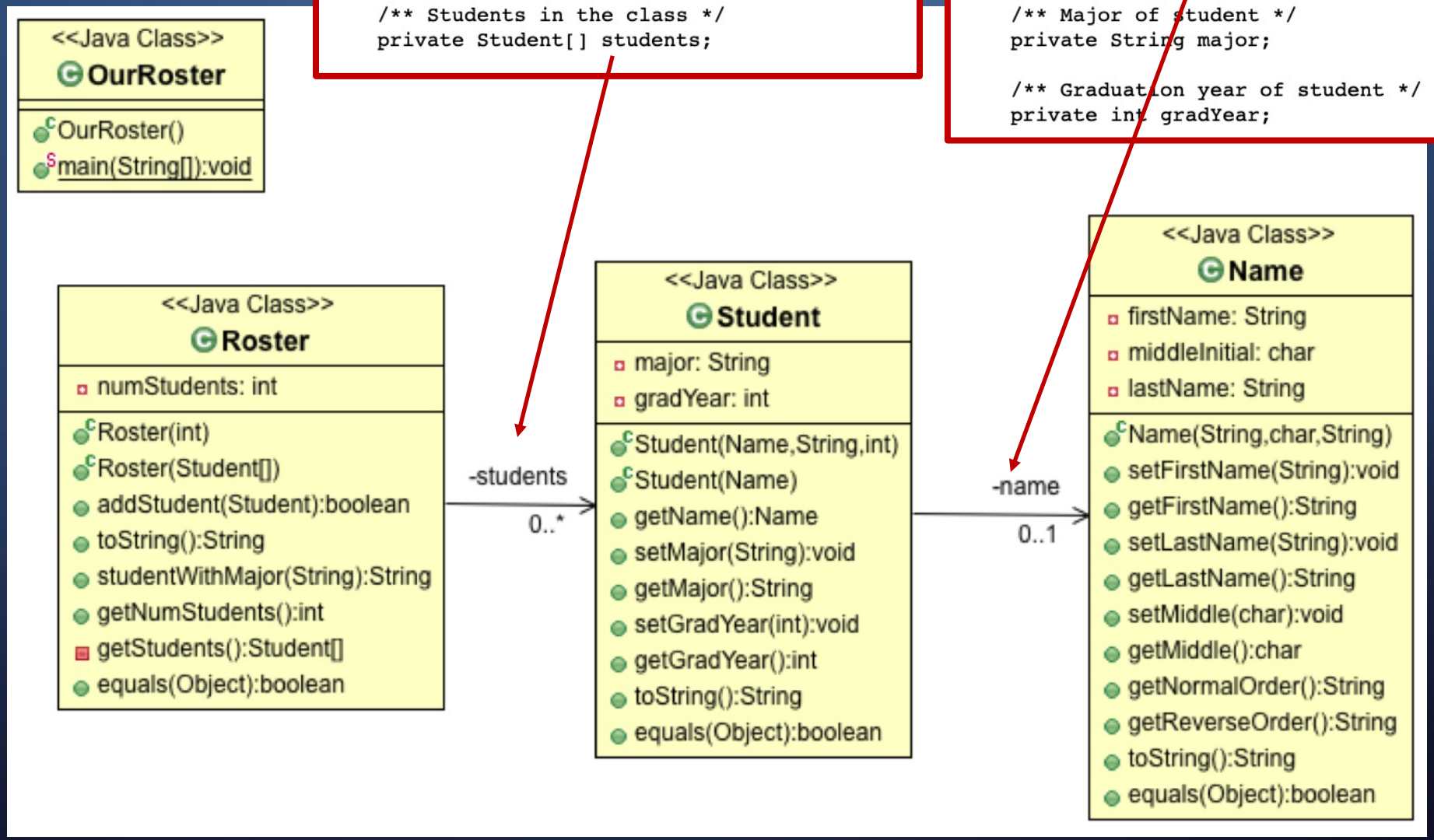
PRINCIPLES OF OBJECT-ORIENTED PROGRAMMING

- Problem Statement
  - Need to make important design decisions prior to attempting to implement
  - "Usually, there is no single best set of classes for a particular application, and hence you should not look for the 'right answer'; however, some designs are better than others."
  - "Program design is not linear, it is iterative."
- Simple Design Process
  - Determine the classes
  - Determine the responsibilities of each class
  - Determine the interactions and collaborations among the classes

- Determine the Classes
  - "Just as a noun is a person, place, or thing, so is an object."
  - Begin by noting the nouns in the problem statement.
  - These nouns give us a good starting point for considering possible classes.
    - Not all nouns will become classes
    - Not all classes will correspond to nouns of problem statement.
- Determine Responsibilities of Each Class
  - "As the nouns indicate classes, the verbs of the problem statement help determine class responsibilities."
  - Consider:
    - What service does the class provide?
    - What is each class's responsibility?
    - What are the actions and behaviors of each class?
    - What attributes/fields?

```
public class Roster {

    /** Current number of students in class */
    private int numStudents;

    /** Students in the class */
    private Student[] students;
```

```
public class Student {

    /** Name of student */
    private Name name;

    /** Major of student */
    private String major;

    /** Graduation year of student */
    private int gradYear;
```



Code: https://go.ncsu.edu/csc116-interacting-classes