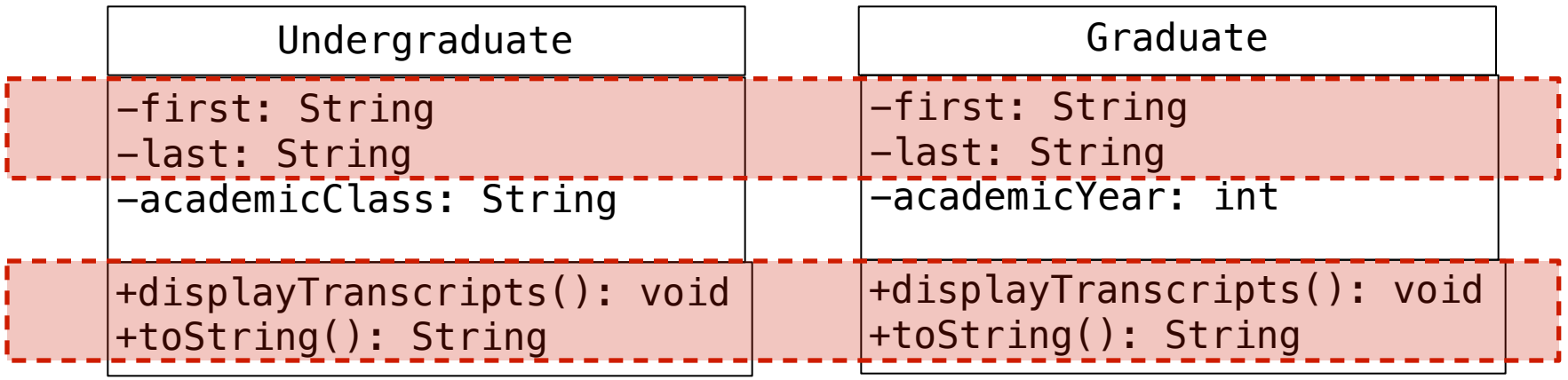
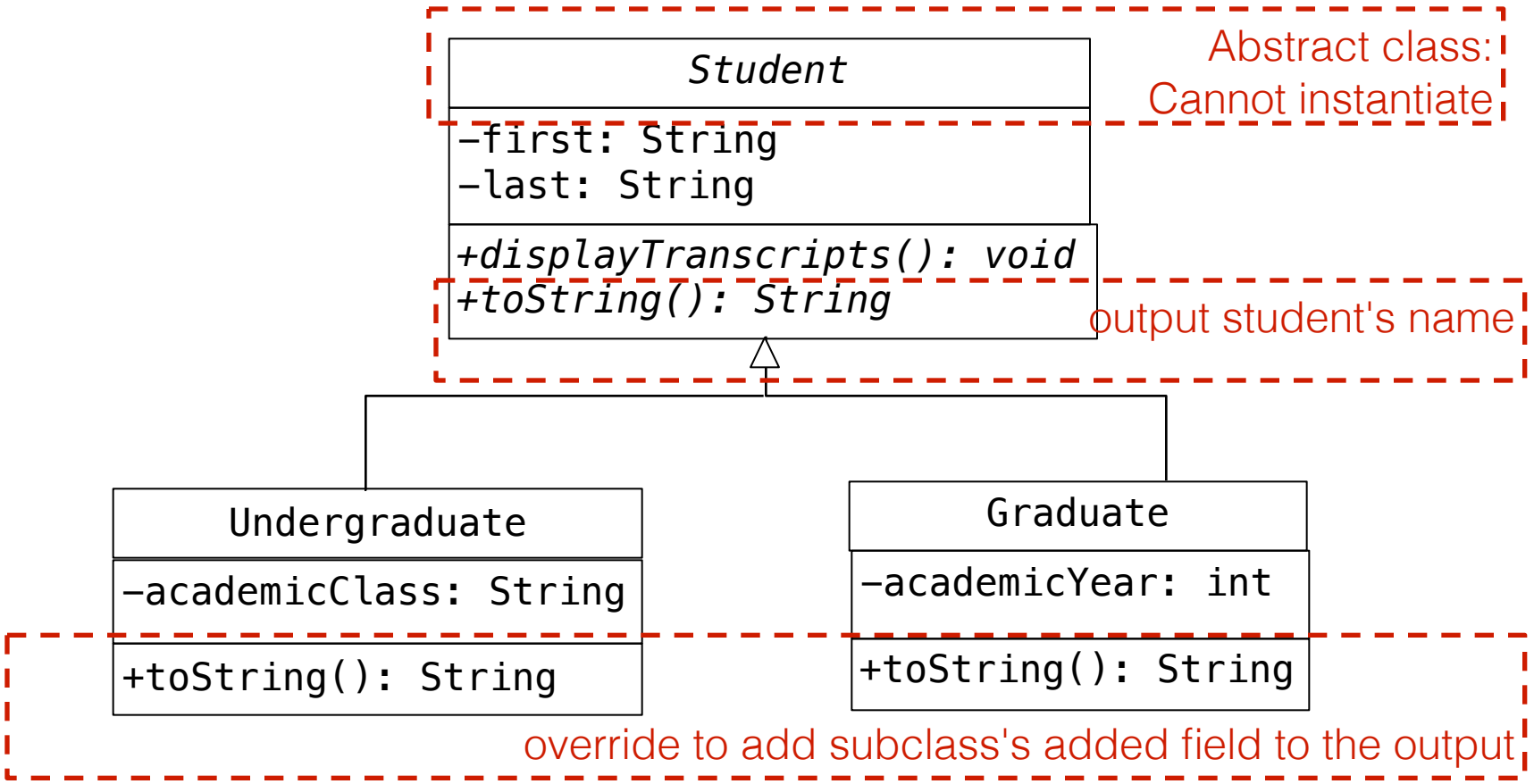
# Ch 11

Inheritance and Polymorphism

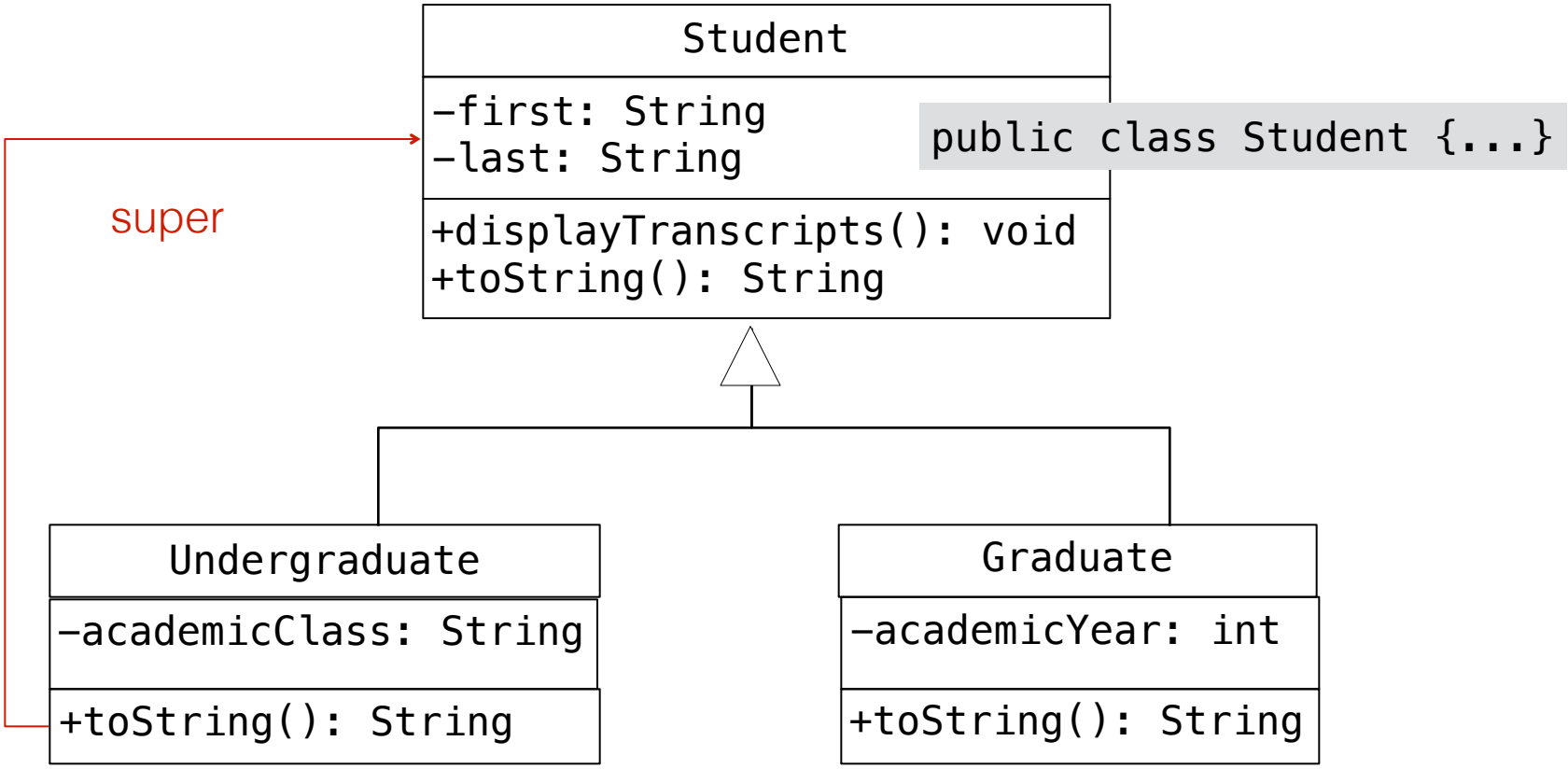
# Use inheritance to factor out common functionality



# Factor common features into `Student`



# `super` gives you access to the superclass's members




```
public class Undergraduate extends Student {...}
public class Graduate extends Student {...}
```

# A subclass constructor calls a superclass constructor

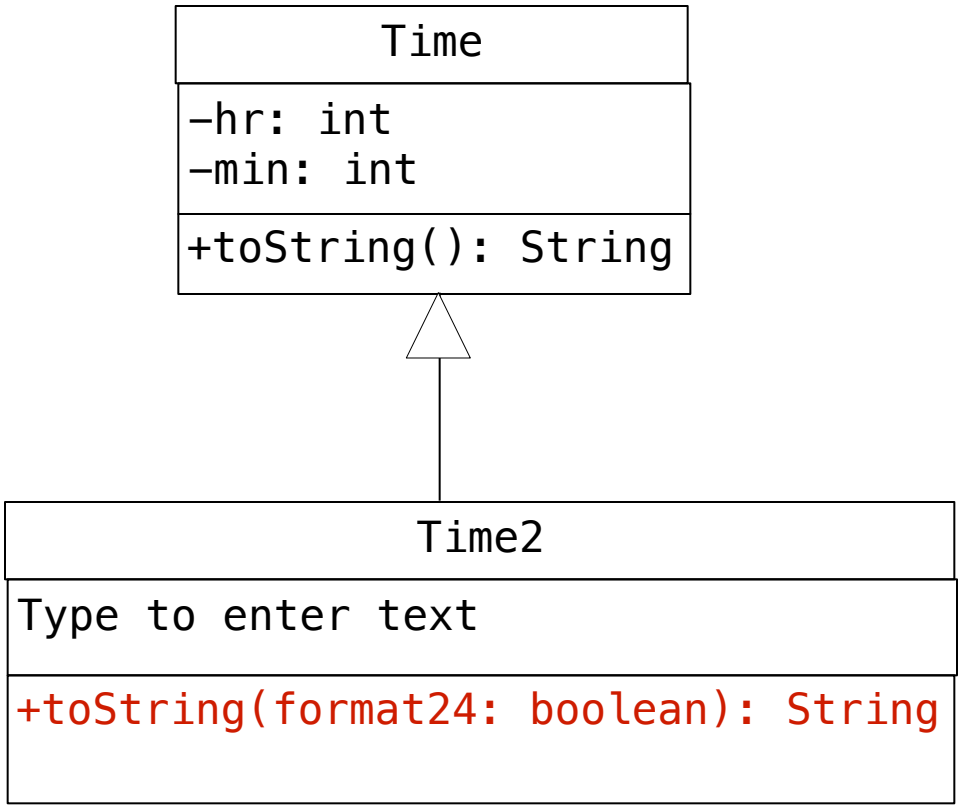
```
 // In subclass Undergraduate

 public Undergraduate() {
    this("unsigned", "unsigned", "unsigned");
 }
```

```
// In subclass Undergraduate

public Undergraduate(String first,
                     String last, String academicClass) {
   // Calls super() implicitly if no super call made
   super(first, last);
   this.academicClass = academicClass;
}
```

# Override a method when you want to do your own thing

```
// In subclass Undergraduate

public String toString() {
    return super.toString() +
           "\nAcademic class: " +
           academicClass;
}
```

# Overload a method when you want to add flexibility



# Polymorphism is build into the Java psychic

```
...
processStudent(new Undergraduate("Allen", "Apple", "Freshman"));
procesStudent(new Graduate("Bill", "Broccoli", 1));


public void processStudent(Student student) {
   System.out.println(student.toString());
}
```

Q? Which toString() method gets executed? In the superclass or subclass?

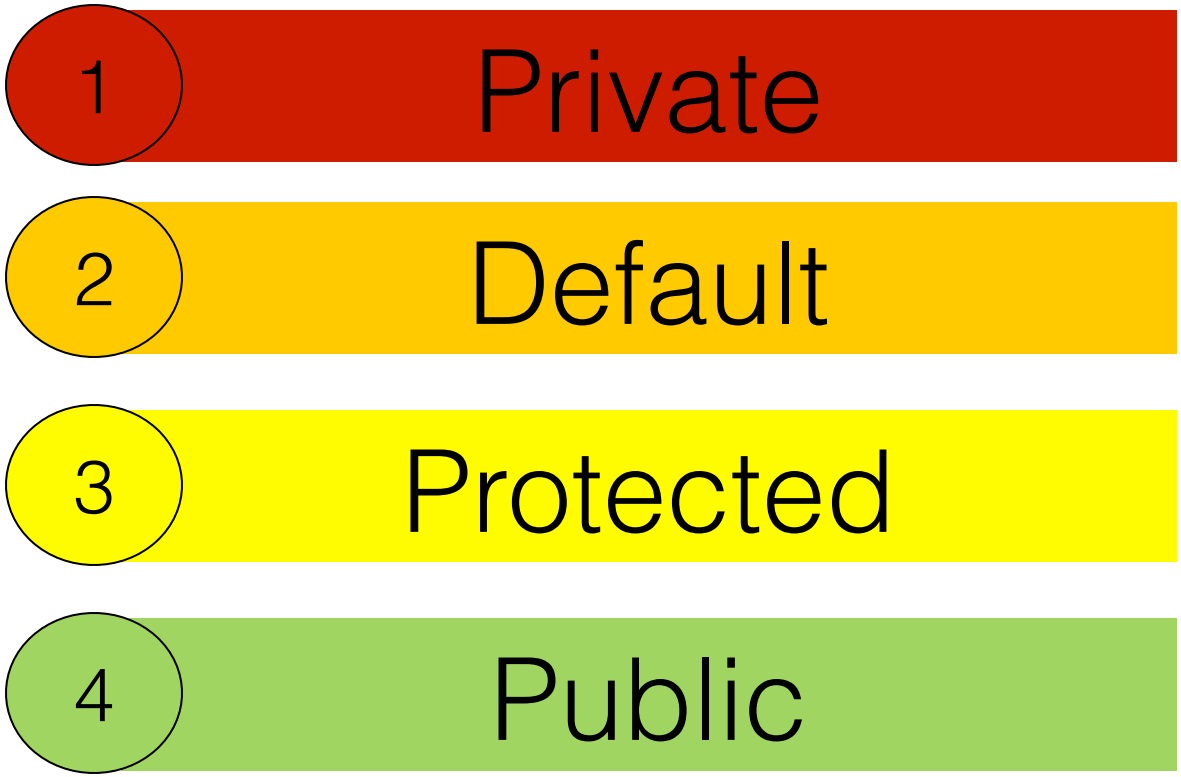
A: Through polymorphism the subclass method gets executed. Here the object's type (Undergraduate or graduate) dictates, rather than the object's reference type (Student)

Dynamic binding means a called method is chosen at run time

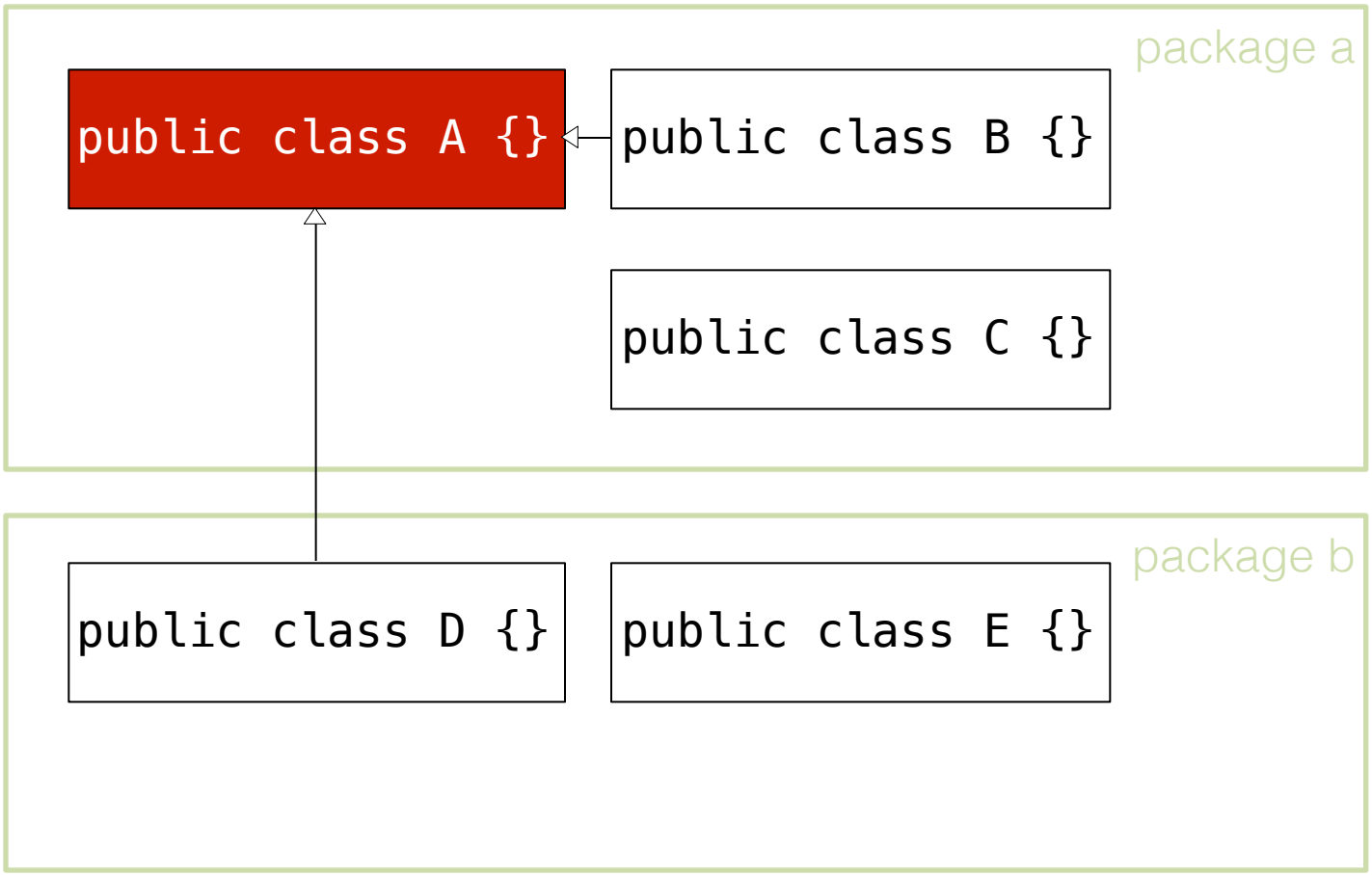
# `instanceof` allows you to find out the object's type at run time

```
public void processStudent(Student student) {
   if (student instanceof Undergraduate) {
      System.out.print("Processing an undergraduate: ");
      System.out.println(((Undergraduate) student).getAcademicClass());

   } else if (student instanceof Graduate) {
      System.out.print("Processing a graduate: ");
      System.out.println(((Graduate) student).getAcademicYear());
   }
}
```

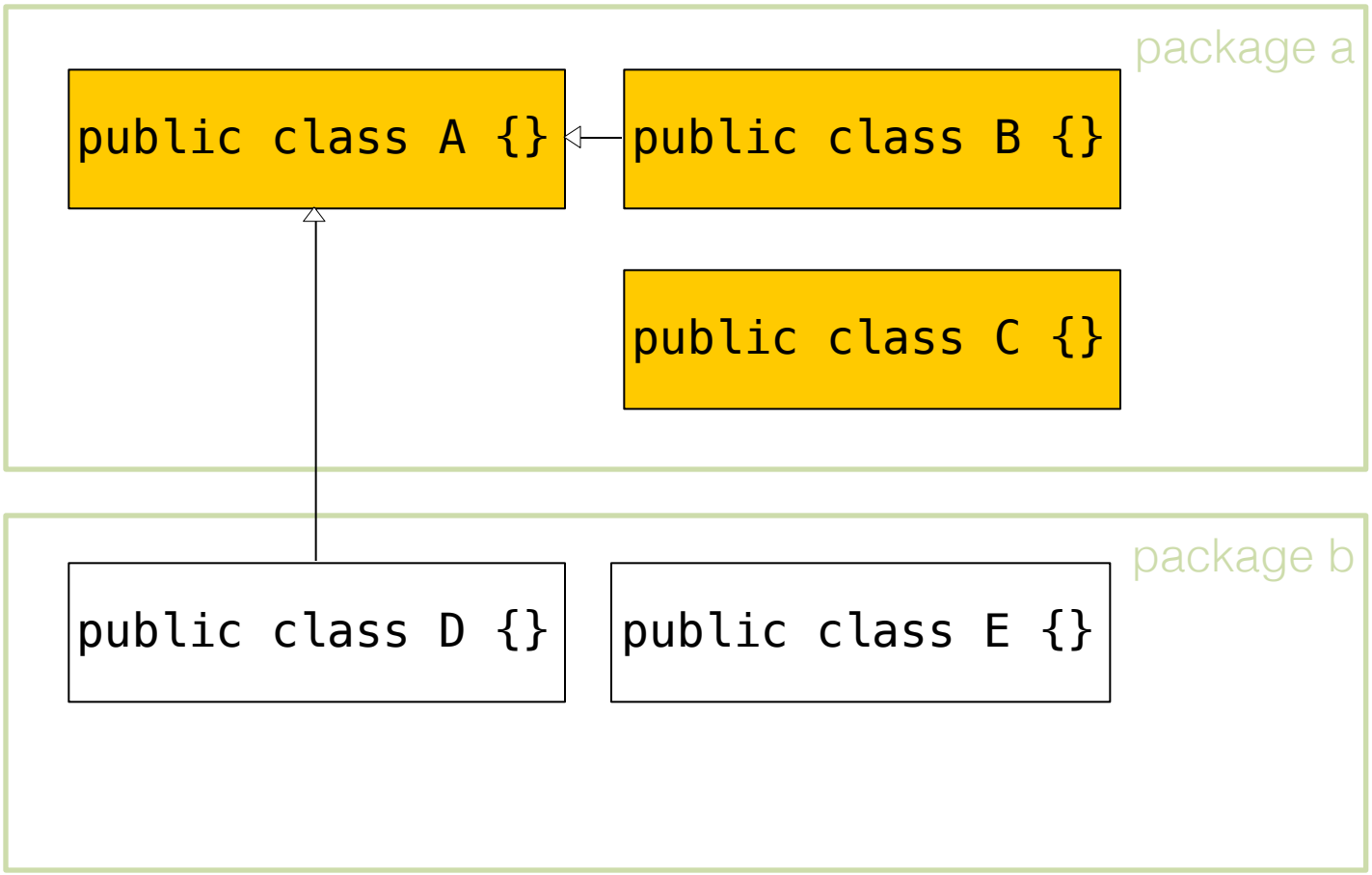
# There are four accessor modes in Java



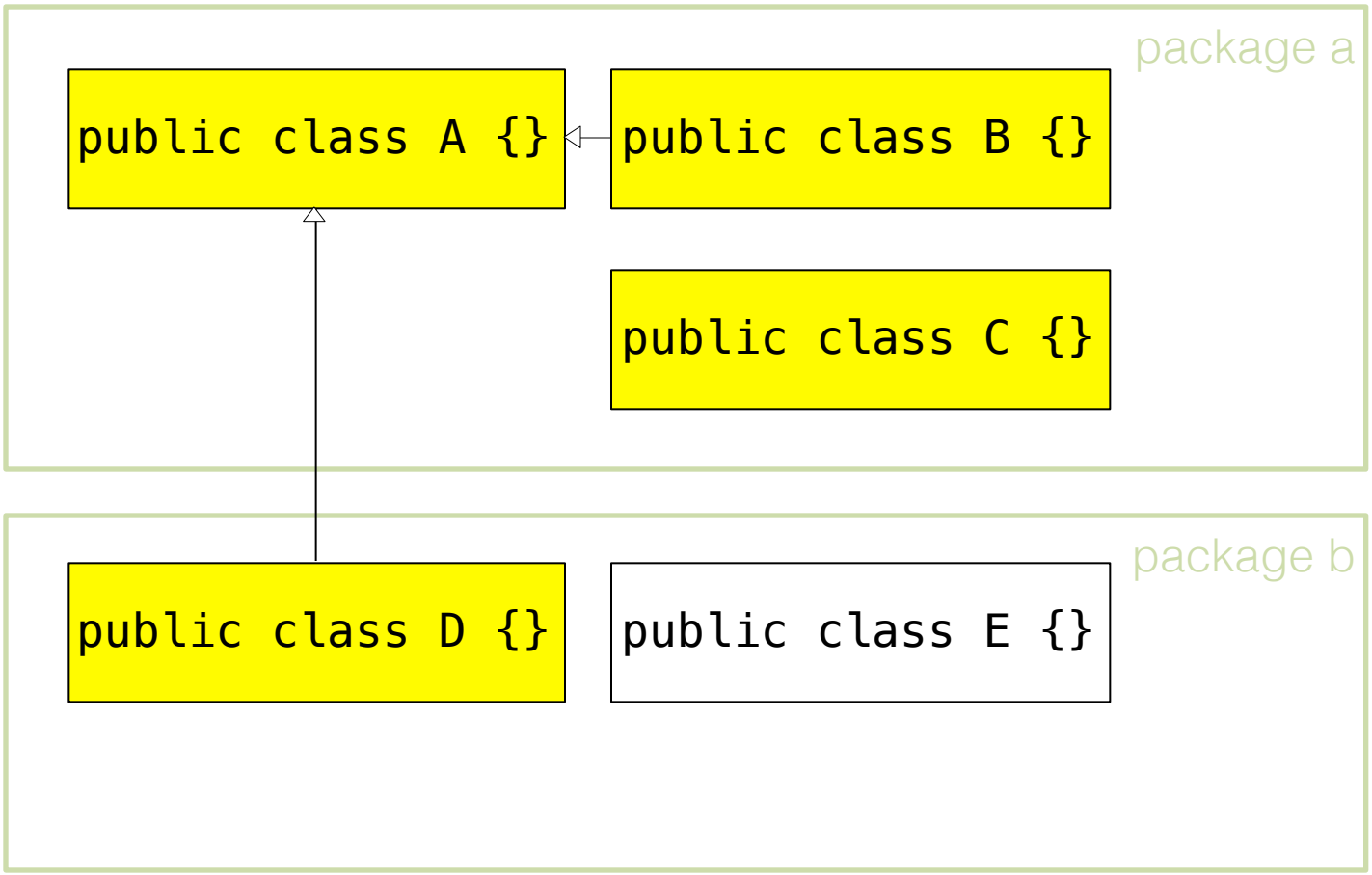
# Private member in A accessible only in defining class



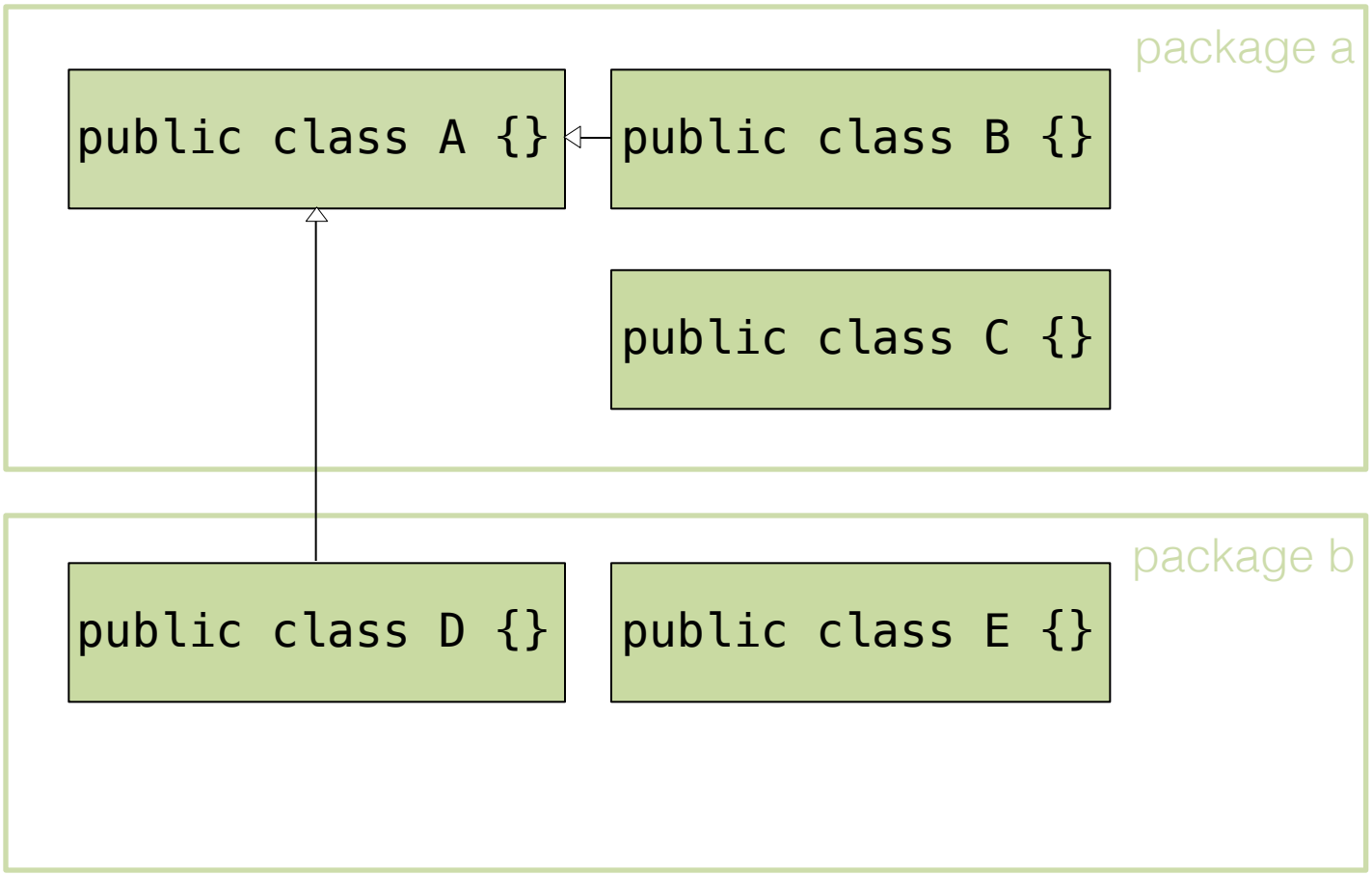
# Default member in A accessible in same package



# Protected member in A accessible in same package and subclasses



# Public member in A accessible by all



# To prevent extending a class use `final`

```
public final class classThatCantBeExtended {
}
```

# To prevent overriding a method use final

```
public final void methodThatCantBeOverriden() {
}
```