Inheritance

The `extends` keyword allows you to define a new class based on an existing class. That way, you can define similar types of objects without having to copy and paste source code.

Manager: Recorder:

Presenter: Reflector:

Content Learning Objectives

After completing this activity, students should be able to:

- Summarize uses of the keywords `extends` and `super`.
- Explain which methods can be called by a variable using polymorphism.
- Predict which method will actually run when polymorphism is involved.

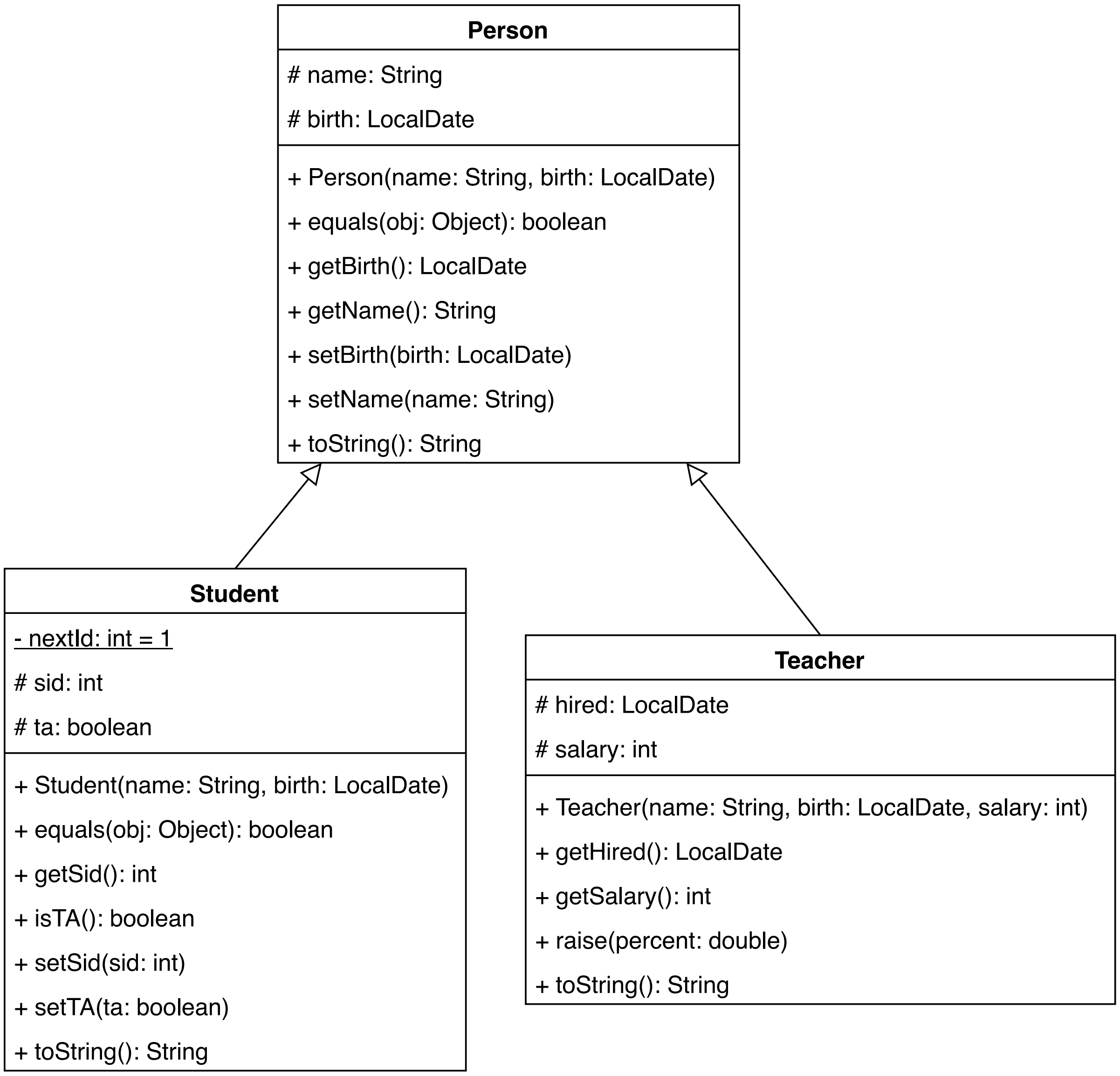
Process Skill Goals

During the activity, students should make progress toward:

- Making conclusions based on IDE hints and program output. (Critical Thinking)

Model 1 Students and Teachers

Answer the following questions by comparing the UML diagram with the provided code (see *Person.java*, *Student.java*, and *Teacher.java*).



Questions (20 min) Start time:

1. Compare *Person.java* with the UML diagram.

a) What keyword does the # symbol represent in UML?

b) Which attributes does the `equals` method compare?

c) What is the `toString` format of a `Person` object?

2. Compare *Teacher.java* with the UML diagram.

a) What keyword does the triangle arrow represent in UML?

b) What keyword is used in the constructor and `toString`?

c) What is the `toString` format of a `Teacher` object?

Read out loud:

> *The code "`Teacher extends Person`" defines an inheritance relationship between the `Teacher` and `Person` classes. `Teacher` is the subclass, and `Person` is the superclass.*
>
> *The keywords `this` and `super` both refer to the same object. However, the data type of `this` is the subclass, and the data type of `super` is the superclass.*

3. The `extends` keyword "copies" the attributes and methods (except for constructors) from the superclass into the subclass. If the same method is defined in both classes, the subclass overrides the superclass. Based on the UML, list *all* attributes and methods of a `Teacher` object.

a) Attributes:

b) Getter methods:

c) Setter methods:

d) Other methods:

4. For the following snippets in *Student.java*, which method of which class is being invoked?

a) `super(name, birth)`

b) `super.equals(obj)`

c) `super.toString()`

5. Fill in each blank with either "is a" or "has a":

a)	`Person`	`String`	d)	`Student`	`String`
b)	`Person`	`LocalDate`	e)	`Teacher`	`Person`
c)	`Student`	`Person`	f)	`Teacher`	`LocalDate`

6. Explain the difference between “is a” and “has a” in the previous question.

7. Why would saying “`Person` is a `Student`” be incorrect? Give a counterexample.

8. Given the following objects,

```
Person p = new Person("Alan", LocalDate.parse("1912-06-23"));
Student s = new Student("Grace", LocalDate.parse("1906-12-09"));
Teacher t = new Teacher("John", LocalDate.parse("1903-12-28"), 123456);
```

which `equals` method of which class will be invoked by the statements below? Explain your reasoning based on the applicable inheritance relationship.

a) `p.equals(p)`

b) `s.equals(s)`

c) `t.equals(t)`

Read out loud:

In Java, every class `extends` exactly one superclass. If no superclass is specified, a class automatically `extends Object`. For example, the `Person` class does not use the `extends` keyword, so the superclass of `Person` is `Object`. The `Object` class provides the default implementation of `equals` and `toString`, among other methods.

Model 2 Variable vs Object Types

Consider the following program:

```
public static void main(String[] args) {
    Person p1 = new Person("Helen", LocalDate.parse("2000-01-02"));
    Student s1 = new Student("John", LocalDate.parse("2000-03-04"));
    Person poly = new Student("Polly", LocalDate.parse("2000-05-06"));

    System.out.println(p1 instanceof Student);
    System.out.println(s1 instanceof Student);
    System.out.println(poly instanceof Student);
}
```

The output of the program is:

```
false
true
true
```

Questions (25 min) Start time:

9. Complete the table below based on the source code:

Variable	Type of Variable	Type of Object
p1		
s1		
poly		

10. Is the `instanceof` operator based on the variable's type or object's type? Justify your answer based on the program's output (shown above).

11. Predict the result of the following expressions. Then run the code on a computer to check your answers.

a) `p1 instanceof Person`

b) `p1 instanceof Object`

c) `s1 instanceof Person`

d) `s1 instanceof LocalDate`

e) `poly instanceof Person`

f) `poly instanceof Teacher`

12. Review your answer to Quesiton #7. Explain why the following statement is invalid:

```
Student s2 = new Person("Chris", LocalDate.parse("2000-07-08"));
```

13. (Optional) Open *Model2.java* in your editor. Answer each question by typing the following code in `main` and pressing Ctrl+Space to list possible completions.

a) Which methods can be called on the `s1` variable? `s1.`

b) Which methods can be called on the `poly` variable? `poly.`

14. Identify a method that is only in the `Student` class (and not in the `Person` class).

a) Which method did you choose?

b) Write code that calls that method on `poly`:

c) What would happen if you tried to run that code on a computer?

d) Are the methods that you can call based on the variable's or object's type?

15. Sometimes you need to call a method from the object's class, even though the variable was declared as a different type. Using your example from the previous question, do the following:

a) Write an if-statement that checks if `poly` "is a" `Student` object.

b) Inside the `if` block, declare a new variable of type `Student`. Typecast the `poly` variable, and assign the result to the `Student` variable.

c) Call the `Student` method on the `Student` variable.

16. Where in the source code of *Person.java* do you see this 3-step pattern?

17. In general, explain why the first two steps (the if statement and the typecast) are needed.

18. (Optional) Trace the execution of the following code using a debugger:

```
LocalDate d = LocalDate.parse("1949-01-17");
Object obj = new Teacher("Anita Borg", d, 123456);
System.out.println(obj.toString());
```

a) What type of variable is `obj`?

b) What type of object does `obj` reference?

c) Which version of `toString` (in which class) is invoked first?

d) Which version of `toString` (in which class) is invoked second?

19. Predict which `equals` methods will be called in the following example. (Optional) Trace the code using a debugger to check your answer.

```
Person j = new Student("John", LocalDate.parse("2000-03-04"));
Person m = new Teacher("Mary", LocalDate.parse("2000-09-10"), 100000);
System.out.println(j.equals(m));
System.out.println(m.equals(j));
```

20. Discuss the following questions. Justify your answers using examples from today's activity.

a) Does the variable's type or object's type determine which methods can be called?

b) Does the variable's type or object's type determine which method is actually called?