Polymorphism

"The dictionary definition of *polymorphism* refers to a principle in biology in which an organism or species can have many different forms. This principle can also be applied to object-oriented programming." (The Java Tutorials)

Manager:		Recorder:	
Presenter:		Reflector:	

Content Learning Objectives

After completing this activity, students should be able to:

- Identify whether two classes have an "is a" or "has a" relationship.
- 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:

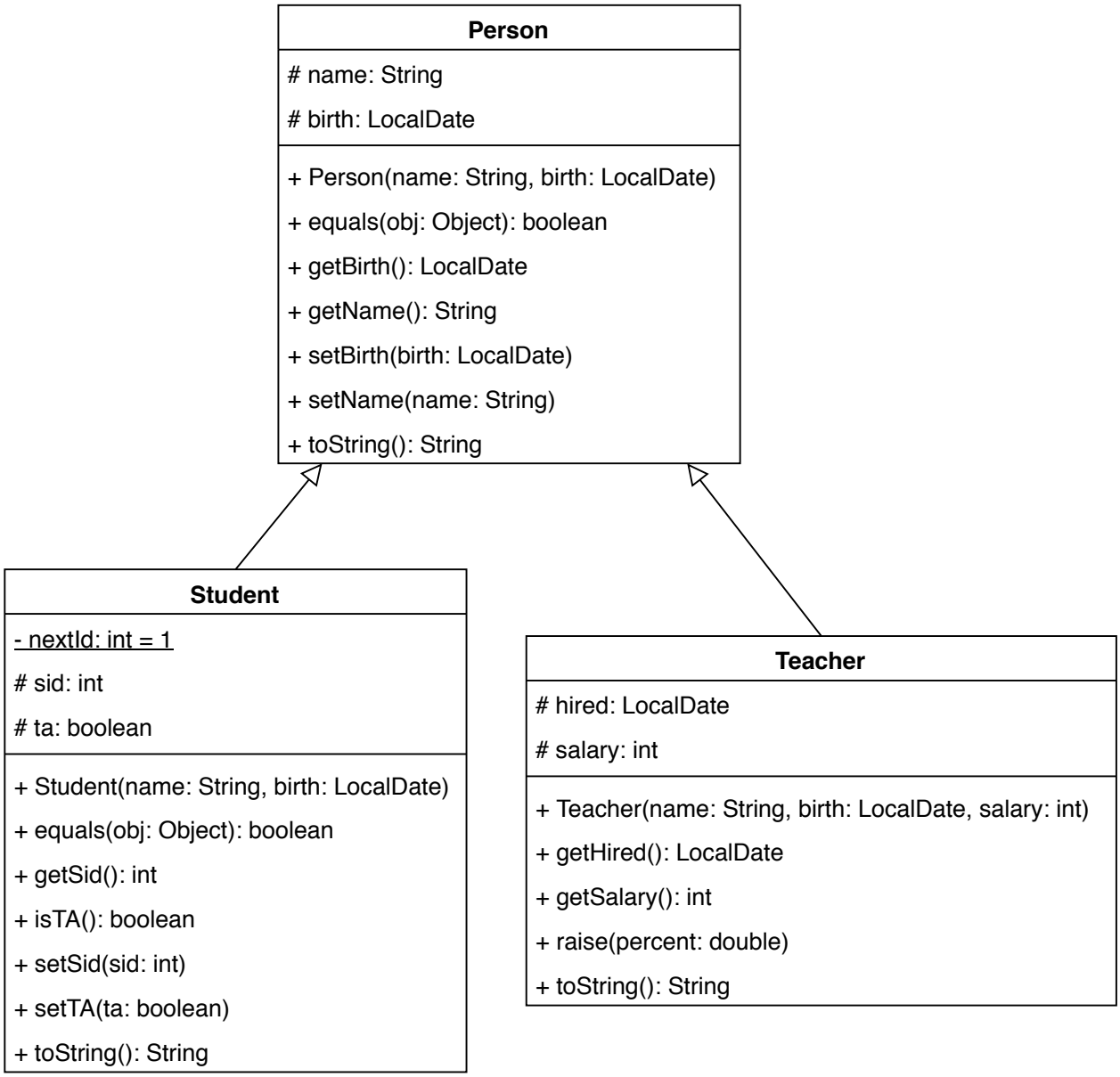
- Verifying program behavior by tracing code with a debugger. (Critical Thinking)

Facilitation Notes

Key questions: #4, #15, #18

Source files: *Model1.java*, *Model2.java*, *Person.java*, *Student.java*, *Teacher.java*

Model 1 Students and Teachers



Questions (15 min) Start time:

1. Based on the UML diagram:

a) What attributes does a `Student` object have? name, birth, sid, ta

b) What attributes does a `Teacher` object have? name, birth, hired, salary

c) Which methods does `Student` override? equals and toString

d) Which methods does `Teacher` override? toString

2. Based on the UML diagram:

a) Which methods does a `Student` and a `Teacher` have in common? (i.e., inherited)

getBirth, getName, setBirth, setName

b) Which methods does a `Student` object have that a `Teacher` object does not have?

getSid, isTA, setSid, setTA

c) Which methods does a `Teacher` object have that a `Student` object does not have?

getHired, getSalary, raise

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

a) `Person` has a `String`

b) `Person` has a `LocalDate`

c) `Student` is a `Person`

d) `Student` has a `String`

e) `Teacher` is a `Person`

f) `Teacher` has a `LocalDate`

4. Explain the difference between "is a" and "has a" in the previous question.

"Is a" refers to inheritance; every subclass is an instance of its parent class. "Has a" refers to composition; an object's attributes may reference other objects.

5. Why would it be incorrect to say "`Person` is a `Student`"?

Inheritance relationships are one-way. Every `Student` is a `Person`, but that doesn't make every `Person` a `Student`.

6. Which `equals` method (in which class) will be invoked by the following code? Explain your reasoning based on the applicable "is a" or "has a" relationship.

```
LocalDate d = LocalDate.parse("1949-01-17");
Teacher t1 = new Teacher("Anita Borg", d, 123456);
Teacher t2 = new Teacher("Anita Borg", d, 123456);
System.out.println(t1.equals(t2));
```

It will invoke the `equals` method in the `Person` class. There is no `equals` method in the `Teacher` class, but `Teacher` is a `Person`. So `Teacher` inherits the `equals` method of `Person`.

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 (30 min) Start time:

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

Variable	Type of Variable	Type of Object
p1	Person	Person
s1	Student	Student
poly	Person	Student

8. Is the `instanceof` operator based on the variable's type or object's type? Justify your answer with a specific example from the program.

It's based on the object's type. The 3rd line of output is true, because poly is a Student, even though it's declared as a Person.

9. Predict the result of the following expressions. Then run the code on a computer to check your answers. Note: `instanceof Object` is always true.

a) `p1 instanceof Person` true

b) `p1 instanceof Object` true

c) `s1 instanceof Person` true

d) `s1 instanceof LocalDate` false

e) `poly instanceof Person` true

f) `poly instanceof Teacher` false

10. Review your answer to Quesiton #5. Explain why the following statement is invalid:

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

The variable's type has to be a parent of the object's type (if not the same). Not all Person objects are Students.

11. 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.`

All the methods of Object, Person, and Student.

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

Only the methods of Object and Person.

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

a) Which method did you choose? getSid, isTA, setSid, or setTA

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

```
poly.setSid(1234);
```

c) What happens when you try to run that code on a computer?

Compiler error: "The method setSid(int) is undefined for the type Person"

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

13. 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 a `Person` variable "is a" `Student` object.

```
if (poly instanceof Student) {
```

b) Inside the if-statement block, declare a new variable of type `Student`. Type-cast the `Person` variable, and assign the result to the `Student` variable.

```
Student stu = (Student) poly;
```

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

```
stu.setSid(1234);
```

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

In the `equals` method.

15. In general, explain why the first two steps (the if statement and type cast) are needed.

The first step is needed to avoid getting a `ClassCastException`. The second step is needed in order to access fields/methods of the subclass.

16. 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`? Object

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

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

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

17. Predict which `equals` methods will be called in the following example. Then 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));
```

In the first `println`, `Student.equals` is called because `j` is a `Student`. (Then `Person.equals` is called via `super`.) In the second `println`, `Person.equals` is called because `m` is a `Teacher`, which inherits `equals` from `Person`.

18. 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?

The variable's type. When the compiler sees a variable, it makes sure the methods are defined in its class (or parent's class).

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

The object's type. This behavior is desired when methods are overloaded: we want to call the method specific to that object.