Designing Objects

Today, we look at how to create and apply custom object types in Java. We will also see a useful data structure called a HashMap.

Manager: Recorder:

Presenter: Reflector:

Content Learning Targets

After completing this activity, you should be able to say:

- Explain the purpose of constructor, accessor, and mutator methods.
- Implement the equals and toString methods for a given class design.
- Design a new class (UML diagram) based on a general description.
- Explain what a HashMap represents.
- List and apply essential HashMap operations.
- Describe the benefits and limitations of HashMaps.

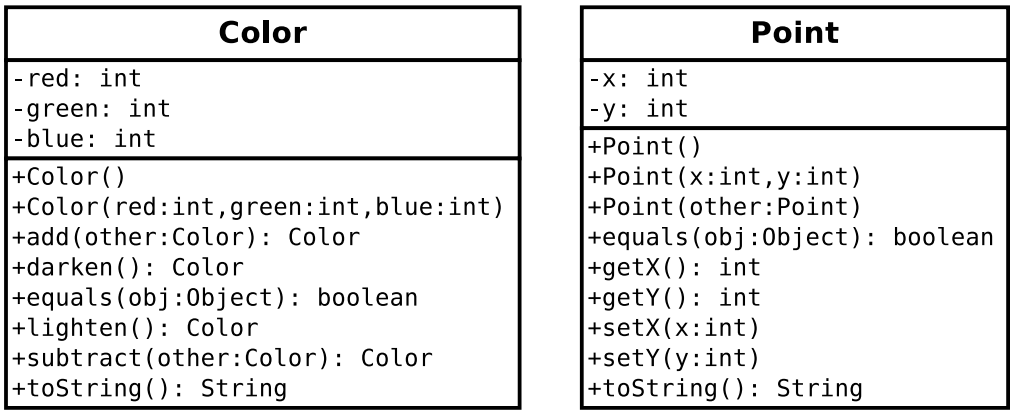
Process Skill Goals

During the activity, you should make progress toward:

- Identifying attributes and data types that model a real-world object. (Problem Solving)

Model 1 Common Methods

Classes are often used to represent abstract data types, such as `Color` or `Point`:



As shown in the UML diagrams, classes generally include the following kinds of methods (in addition to others):

- **constructor** methods that initialize new objects
- **accessor** methods (getters) that return attributes
- **mutator** methods (setters) that modify attributes

Questions (15 min) Start time:

1. Identify the constructors for the `Color` class. What is the difference between them?

2. What kind of constructor does the `Point` class have that the `Color` class does not?

3. Identify an accessor method in the `Point` class.

a) What is the name of the method?

b) Which instance variable does it get?

c) What arguments does the method take?

d) What does the method return?

4. Identify a mutator method in the `Point` class.

a) What is the name of the method?

b) Which instance variable does it set?

c) What arguments does the method take?

d) What does the method return?

5. How would you define accessor methods for each attribute of the `Color` class? Write your answer using UML syntax.

6. How would you define mutator methods for each attribute of the `Color` class? Write your answer using UML syntax.

7. The `Color` class does not provide any accessors or mutators. Instead, it provides methods that return new `Color` objects. Why do you think the class was designed this way?

Model 2 Object Methods

In addition to providing constructors, getters, and setters, classes often provide `equals` and `toString` methods. These methods make it easier to work with objects of the class.

As a team, review the provided *Color.java* and *Point.java* files. Run each program to see how it works. Then answer the following questions using the source code (don't just guess).

Questions (15 min) Start time:

8. Based on the output of *Color.java*, what is the value of each expression below?

```
Color black = new Color();
Color other = new Color(0, 0, 0);
Color gold = new Color(255, 215, 0);
```

a) `black == other`

b) `black == gold`

c) `black.toString()`

d) `black.equals(other)`

e) `black.equals(gold)`

f) `gold.toString()`

9. What is the purpose of the `toString` method?

10. Based on the output of *Point.java*, what is the value of each expression below?

```
Point p1 = new Point();
Point p2 = new Point(0, 0);
Point p3 = new Point(3, 3);
```

a) `p1 == p2`

b) `p1.toString()`

c) `p3.toString()`

d) `p1.equals(p2)`

e) `p1.equals("(0, 0)")`

f) `p3.equals("(3, 3)")`

11. What is the purpose of the `equals` method?

12. Examine *Point.java* again. What is the purpose of the `if`-statement in the `equals` method?

13. How could you modify the `equals` method to cause both #10e and #10f to return `true`?

Model 3 Shelter Animal

Classes often represent objects in the real world. In this section, you will design a new class that represents an `Animal` in a shelter like the one below. This will be the first in a series of activities in which we build up a record-keeping system for an animal shelter.

Photo by Anna Kumpan on Unsplash

Questions (15 min) Start time:

14. Identify four or more attributes (a.k.a. fields) that should be in the `Animal` class. For each attribute, indicate what data type would be most appropriate.

15. Using UML syntax, define two or more constructors for the `Animal` class.

16. Define two or more accessor methods for the `Animal` class. Include arguments and return values, using the same format as a UML diagram.

17. Define two or more mutator methods for the `Animal` class. Include arguments and return values, using the same format as a UML diagram.

18. Describe how you would implement the `equals` method of the `Animal` class.

19. Describe how you would implement the `toString` method of the `Animal` class.

20. When constructing (or updating) an `Animal` object, which arguments would you need to validate? What are the valid ranges of values for each attribute?

Model 4 HashMaps

Questions (25 min) Start time:

We will often find useful a ***map*** data structure, which associates ***keys*** with ***values***. A key is some type of unique identifier, and a value is an object containing the data attached to that identifier. Each key-value pair is called an ***entry*** in the map. In some languages, a map is called a dictionary; keys are like words in a dictionary, and the values are like definitions.

21. Suppose we have a `Student` class and we want to create a map that lets us look up a student by their (String) username. Which should be the key type, and which should be the value?

The built-in Java map class that we will use is `HashMap`. Behind the scenes, `HashMap` uses a technique called hashing to speed up its operations. You will learn more about hashing in CSSE 230, but for now, just know that HashMap lookups are fast. We declare/initialize HashMaps like we do for ArrayLists, but with two type parameters (key, then value), instead of one.

22. Complete the following declaration and initialization of the student-username map.

```
HashMap<           ,           > usernameToStudent = new HashMap<>();
```

23. Find the HashMap class docs in the Java API. Looking at its methods, how would we…

add the new entry `username` → `student`?

check if the map already has `username`?

check if the map already has `student`?

determine whether the map has no entries?

retrieve the Student for `username`, if it exists?

You will encounter other useful HashMap methods in a programming assignment. We'll now see how HashMaps can accelerate object lookups based on identifiers.

24. Review *AnimalShelterMain.java*. How does this class currently store its `Animal` objects?

25. Examine the methods `getAnimalbyName` & `updateAnimalWeight`. What flaw(s) do you see?

26. Replace the `animals` field of type `ArrayList<Animal>` with two HashMap fields: `animalsByID` and `animalsByName`. What should the key and value types be for these HashMaps?

27. Update the constructor, `addAnimal`, `getAnimalbyName`, and `updateAnimalWeight` methods to use the new fields. After all changes, run the program again. How has the output changed?

28. What is one potential downside to using the second `HashMap`, `animalsByName`?