

Keep What Matters

Abstraction and program design



Same phone, two descriptions

PAUSE AND TRY

Describe your phone in three facts, twice.

FOR SOMEONE WHO MIGHT BUY IT

1. _____
2. _____
3. _____

FOR THE PERSON REPAIRING IT

1. _____
2. _____
3. _____

Which facts made both lists? Which made only one?

By the end of class you can...

- 1 Explain abstraction: keep the details a program needs, hide the rest.
- 2 Tell what data means apart from how it is stored (data abstraction).
- 3 Sort a class's attributes into instance variables and class variables.
- 4 Use procedural abstraction: name steps as methods, generalize with parameters.
- 5 Plan a class in words or a class diagram before coding it.

1

PART 1

Abstraction: keep what matters

Which details matter depends on the program



Abstraction reduces complexity by hiding details that do not matter for the problem and keeping the ones that do.

Fact about a dog	Vet clinic program	Dog show program
Weight	Keep	Hide
Vaccination dates	Keep	Hide
Breed	Keep	Keep
Judges' scores	Hide	Keep
Favorite toy	Hide	Hide

One idea, two ways to store it

ONE INT: SECONDS

```
int lengthInSeconds = 215;  
int mins = lengthInSeconds / 60;  
int secs = lengthInSeconds % 60;  
System.out.println(mins + ":" + secs);
```

OUTPUT

3:35

TWO INTS: MINUTES AND SECONDS

```
int mins = 3;  
int secs = 35;  
System.out.println(mins + ":" + secs);
```

OUTPUT

3:35



Data abstraction separates what data **means** (a song that is 3 minutes 35 seconds long) from how it is **stored** (its representation). A good name, like `lengthInSeconds`, carries the meaning.

Keep or hide?

PAUSE AND TRY

A hiking app helps people pick a trail for today. Which facts should it keep?

Trail name

Length in miles

Elevation gain

Year it was built

Today's weather

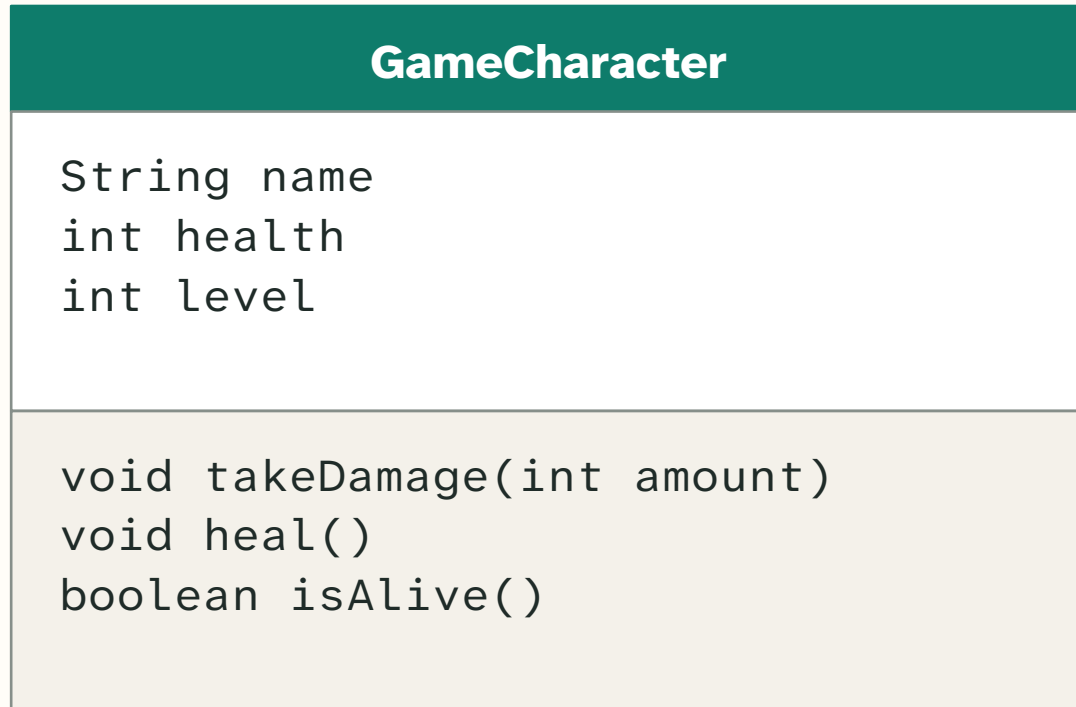
Name of the ranger who
mapped it

A large, bold, teal-colored number '2' is centered within a white rounded square that has a slight 3D effect and a shadow.

PART 2

Attributes and class diagrams

A class diagram shows the plan



Top: the class name

What kind of object this is.

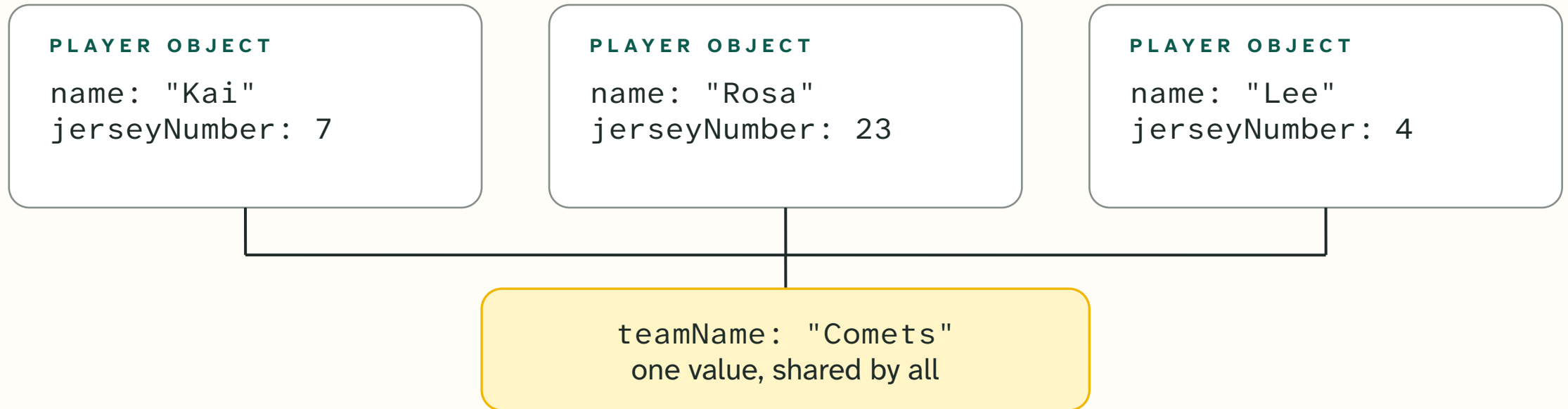
Middle: attributes

Data each object keeps, written as type and name.
Defined in the class, outside any method.

Bottom: behaviors

What an object can do, written as method headers.

Instance variable or class variable?



Instance variable: each object has its own copy. Changing Kai's number changes only Kai.

Class variable: one copy shared by every object. Rename the team once and every player sees it.

"Every object has one" does not make it shared



The test is not "does every object have this?" It is "**should every object have the same value?**"

HEALTH AS A CLASS VARIABLE

The dragon hits Ada for 30. There is only one health value, so Ada, Ben, and Cy all drop by 30 at once.

HEALTH AS AN INSTANCE VARIABLE

The dragon hits Ada for 30. Only Ada's health drops. Ben and Cy keep their own values.

Instance or class?

PAUSE AND TRY

Mark each HotelRoom attribute: instance or class?

Attribute	Instance or class?
<code>int roomNumber</code>	
<code>boolean occupied</code>	
<code>String hotelName</code>	
<code>int checkoutHour</code>	

Every room in this hotel has the same checkout hour. Ask of each attribute: should every room hold the same value?

A large, white, rounded square containing the number '3' in a bold, teal, sans-serif font.

3

PART 3

Procedural abstraction

Break a big job into named steps

INSIDE MAIN: READS LIKE A PLAN

```
double avg = average(90, 80);  
String grade = letter(avg);  
printReport(avg, grade);
```

OUTPUT

```
Average: 85.0  
Grade: B
```

Method decomposition: one method for each smaller behavior, named for what it does.

HELPER METHODS (PRINTREPORT NOT SHOWN)

```
public static double average(int a, int b) {  
    return (a + b) / 2.0;  
}  
  
public static String letter(double avg) {  
    if (avg >= 90) {  
        return "A";  
    } else if (avg >= 80) {  
        return "B";  
    }  
    return "C or below";  
}
```

Copies become one method with a parameter

BEFORE: DUPLICATED CODE

```
int quiz1 = 90 + 5;  
if (quiz1 > 100) {  
    quiz1 = 100;  
}  
int quiz2 = 98 + 5;  
if (quiz2 > 100) {  
    quiz2 = 100;  
}
```

AFTER: ONE GENERAL METHOD

```
public static int curve(int score) {  
    int curved = score + 5;  
    if (curved > 100) {  
        curved = 100;  
    }  
    return curved;  
}
```

CALLS

```
System.out.println(curve(90));  
System.out.println(curve(98));  
System.out.println(curve(61));
```

OUTPUT

```
95  
100  
66
```

A **parameter** lets one method work for many values. Write it once, call it many times: **code reuse**.

Change how it works, not what it promises

NEW INSIDE, SAME SIGNATURE

```
public static int curve(int score) {  
    if (score >= 95) {  
        return 100;  
    }  
    return score + 5;  
}
```

NEW SIGNATURE, OLD CALL CURVE(90)

```
public static int curve(int score, int add) {  
    return score + add;  
}
```

Safe: same results, callers never notice

OUTPUT

```
Curve.java:3: error: method curve in class  
Curve cannot be applied to given types;  
    required: int,int  
    found:    int
```



Callers depend on the **signature** and on **what the method does**. Keep both, and the inside can change freely.

Tell the callers, or not?

PAUSE AND TRY

Which changes to `curve` are safe without telling callers?

A Shorter body, no cap

```
return score + 5;
```

B Rename the method

```
addCurve(int score)
```

C Rewrite the body

```
int curved = score + 5;  
if (curved >= 100) {  
    return 100;  
}  
return curved;
```

A large, white, rounded square containing a bold, dark teal number '4'.

4

PART 4

Design before you code

From a description to a design

THE DESCRIPTION

Each rental bike has an ID and knows whether it is rented. Every bike charges the same hourly price. A bike can be rented, returned, and can report the cost for a number of hours.

1. Facts each bike remembers become attributes.
2. Same value for every bike? Class variable.
3. Verbs become behaviors; "for a number of hours" means a parameter.
4. Check: does each behavior have the data it needs?

RentalBike

```
String bikeId  
boolean rented  
double hourlyPrice    (class)
```

```
void rent()  
void giveBack()  
double cost(int hours)
```

Words to know

Abstraction

Keep the details a problem needs; hide the rest.

Data abstraction

Use data by its name and meaning, not how it is stored.

Attribute

Data a class keeps, defined outside any method.

Instance vs. class variable

Own value in each object, or one value shared by all.

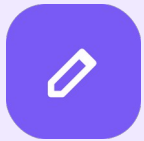
Procedural abstraction

Use a method knowing what it does, not how.

Method decomposition

Break a big behavior into smaller methods.

Plan one before you leave



With a partner, sketch a class diagram for something in this room: two or three attributes, two behaviors.

SENTENCE STARTER

We left out ____ because _____. We made ____ shared because every object _____.

Next: the Keep What Matters puzzle worksheet.