

Exercise walk-through

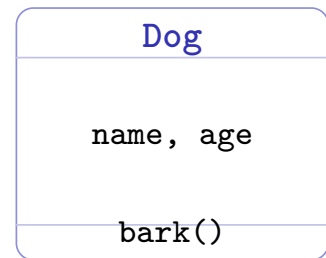
3.9

this Keyword

AP Computer Science A

What we will cover

- 1 Abstraction & program design
- 2 The anatomy of a class
- 3 Constructors & methods
- 4 References & static members
- 5 Scope & the this keyword

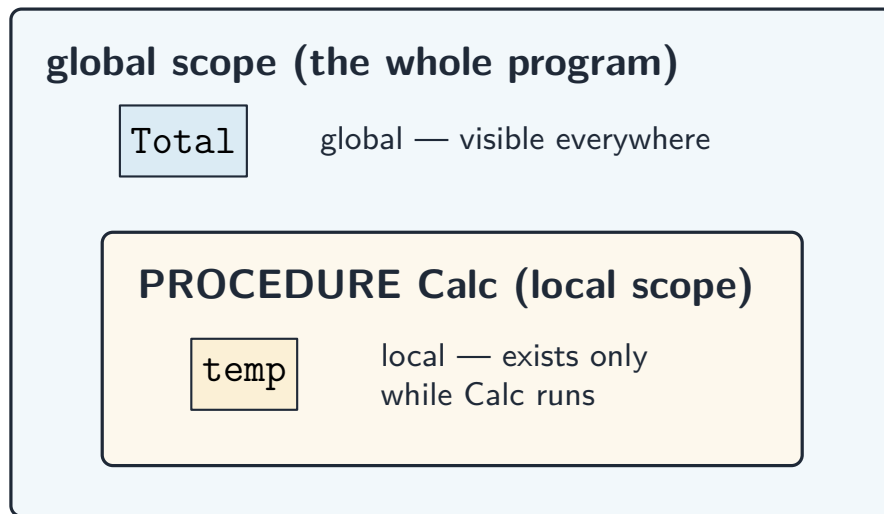


Scope & the this keyword

Scope is where a name is visible.

A local variable lives only inside its method; a field lives as long as the object.

`this` refers to the current object. Use it when a parameter name hides a field: `this.name = name`; sets the field from the parameter.



Check yourself

- 1 Can a method change the caller's object through a reference?
- 2 Can it change which object the caller's variable points to?
- 3 What does `static` mean – per object, or per class?
- 4 What does `this` refer to?

Answers 1. yes 2. no 3. per class – shared by all objects 4. the current object

Exercises 2.1-2.2

2.1 State what `this` refers to while `b.sameAge(a)` runs.

[1]

2.2 A constructor has parameter `int age` and the line `age = age;`. State the object's `age` afterwards, and explain why.

[2]

Exercises 2.1-2.2

2.1 State what `this` refers to while `b.sameAge(a)` runs.

[1]

2.1

- the object `b` refers to

2.2 A constructor has parameter `int age` and the line `age = age;`. State the object's `age` afterwards, and explain why.

[2]

2.2

- `0`
- `age` means the parameter on both sides, so the field is never set and keeps its default

Exercise 2.3

2.3 Write a method `isOlder` in `Pet` that returns `true` if this pet is older than `other`. [2]

Exercise 2.3

2.3 Write a method `isOlder` in `Pet` that returns `true` if this pet is older than `other`. [2]

2.3

```
public boolean isOlder(Pet other)
{
    return this.age > other.age;
}
```

- `boolean` header with a `Pet` parameter
- `this.age > other.age`

Exercises 3.1-3.2

3.1 In `public void setName(String name)`, which line stores the parameter in the field? [1]

☐

A `name = name;`

B `this.name = name;`

C `name = this.name;`

D `this = name;`

3.2 Where is **this** **not** allowed? [1]

☐

A in a constructor

B in an accessor method

C in a **static** method

D in a mutator method

Exercises 3.1-3.2

3.1 In `public void setName(String name)`, which line stores the parameter in the field? [1]

- ☐ A `name = name;`
☐ B `this.name = name;`
☐ C `name = this.name;`
☐ D `this = name;`

3.1 B

- `this.name` is the field

3.2 Where is **this** **not** allowed? [1]

- ☐ A in a constructor
☐ B in an accessor method
☐ C in a **static** method
☐ D in a mutator method

3.2 C

- a **static** method has no current object

Exercise 3.3

3.3 Pet p = new Pet("Mia", 3); What is printed by p.show();?

[1]

```
public void show()
{
    int age = 10;
    System.out.println(age + this.age);
}
```

☐

A 10

B 13

C 3

D 20

Exercise 3.3

3.3 Pet p = new Pet("Mia", 3); What is printed by p.show();?

[1]

```
public void show()
{
    int age = 10;
    System.out.println(age + this.age);
}
```

☐

A 10

B 13

C 3

D 20

3.3 B

- local age 10 plus the field this.age 3

Exercise 3.4 (a)

3.4 A `Team` class has `private ArrayList<Player> players;`. A `Player` has `private Team team;` and a method `join(Team t)`.

(a) Write a constructor for `Player` with parameters `String name` and `int number` that sets fields with the **same names**. [2]

Exercise 3.4 (a)

3.4 A `Team` class has `private ArrayList<Player> players;`. A `Player` has `private Team team;` and a method `join(Team t)`.

(a) Write a constructor for `Player` with parameters `String name` and `int number` that sets fields with the **same names**. [2]

(a)

```
public Player(String name, int number)
{
    this.name = name;
    this.number = number;
}
```

- `this.name = name`
- `this.number = number`

Exercise 3.4 (b)

3.4 A `Team` class has `private ArrayList<Player> players;`. A `Player` has `private Team team;` and a method `join(Team t)`.

(b) `join` should set the player's `team` to `t` and add the player to that team with `t.add Player(...)`. Write `join`. [3]

Exercise 3.4 (b)

3.4 A `Team` class has `private ArrayList<Player> players;`. A `Player` has `private Team team;` and a method `join(Team t)`.

(b) `join` should set the player's `team` to `t` and add the player to that team with `t.addPlayer(...)`. Write `join`. [3]

(b)

```
public void join(Team t)
{
    team = t;
    t.addPlayer(this);
}
```

- `void` header with a `Team` parameter
- sets `team` to `t`
- passes `this` to `addPlayer`

Exercise 3.4 (c)

3.4 A `Team` class has `private ArrayList<Player> players;`. A `Player` has `private Team team;` and a method `join(Team t)`.

(c) Explain what the argument you passed to `addPlayer` refers to.

[1]

Exercise 3.4 (c)

3.4 A `Team` class has `private ArrayList<Player> players;`. A `Player` has `private Team team;` and a method `join(Team t)`.

(c) Explain what the argument you passed to `addPlayer` refers to. [1]

(c)

- `this` refers to the player object whose `join` method is running