



Course Progress

Course > Readings/Videos > Reading 9: Abstract Data Types > Questions

 Previous























Next 

## Questions

 Bookmark this page

### access control

1/1 point (graded)

In order to understand abstract data types, you will need to be comfortable with access control in Java -- the `public` and `private` attributes. If you have trouble with these questions, you may want to start with this week's Java Tutor assignment and then come back to this reading.

The following questions about access control use the code below. Study it first, then answer the questions.

```
class Wallet {
    private int amount;

    public void loanTo(Wallet that) {
        // put all of this wallet's money into that wallet
        /*A*/      that.amount += this.amount;
        /*B*/      amount = 0;
    }

    public static void main(String[] args) {
        /*C*/      Wallet w = new Wallet();
        /*D*/      w.amount = 100;
        /*E*/      w.loanTo(w);
    }
}

class Person {
    private Wallet w;

    public int getNetWorth() {
        /*F*/      return w.amount;
    }

    public boolean isBroke() {
        /*G*/      return Wallet.amount == 0;
    }
}
```

Which of the following statements are true about the line marked `/*A*/` ? Check all that apply.

`that.amount += this.amount;`

☒ The reference to `this.amount` is allowed by Java.

☐ The reference to `this.amount` is not allowed by Java because it uses `this` to access a private field.



### Explanation

Private fields and methods can be used by any code in the same class.

Submit

Show Answer

Answers are displayed within the problem

## access control, part 2

1/1 point (graded)

Which of the following statements are true about the line marked `/*A*/` ? Check all that apply.

```
that.amount += this.amount;
```

☒ The reference to `that.amount` is allowed by Java.

☐ The reference to `that.amount` is not allowed by Java because `that.amount` is a private field in a different object.

☐ The reference to `that.amount` is not allowed by Java because it writes to a private field.

☐ The illegal access(es) are caught statically.

☐ The illegal access(es) are caught dynamically.



### Explanation

Wallet's private fields and methods can be used by any code in the Wallet class, even to access private fields in more than one Wallet object, not just in "this". Roughly speaking, any code within the curly braces of Wallet's class body can touch Wallet's private fields and methods. (This isn't strictly true because Wallet might contain nested class definitions that don't automatically get access to Wallet's private fields and methods, but aside from that, it's a useful rule of thumb.)

Submit

Show Answer

Answers are displayed within the problem

## access control, part 3

1/1 point (graded)

Which of the following statements are true about the line marked `/*B*/` ? Check all that apply.

```
amount = 0;
```

☒ The reference to `amount` is allowed by Java.

☐ The reference to `amount` is not allowed by Java because it doesn't use `this`.

☐ The illegal access is caught statically.

☐ The illegal access is caught dynamically.



#### Explanation

Private fields and methods can be used by any code in the same class. For fields, the "this" reference is implicit and can be omitted.

Submit

Show Answer

Answers are displayed within the problem

### access control, part 4

1/1 point (graded)

Which of the following statements are true about the line marked `/*C*/` ? Check all that apply.

```
Wallet w = new Wallet();
```

☒ The call to the `Wallet()` constructor is allowed by Java.

☐ The call to the `Wallet()` constructor is not allowed by Java because there is no public `Wallet()` constructor declared.

☐ The illegal access is caught statically.

☐ The illegal access is caught dynamically.



#### Explanation

In Java, a class with no explicitly-declared constructors gets an automatic public constructor that takes no arguments.

Submit

Show Answer

Answers are displayed within the problem

### access control, part 5

1/1 point (graded)

Which of the following statements are true about the line marked `/*D*/` ? Check all that apply.

```
w.amount = 100;
```

☒ The access to `w.amount` is allowed by Java.

☐ The access to `w.amount` is not allowed by Java because `amount` is private.

☐ The illegal access is caught statically.

☐ The illegal access is caught dynamically.



#### Explanation

`Wallet`'s private fields and methods can be used by any code in the `Wallet` class, even static methods. Roughly speaking, any code within the curly braces of `Wallet`'s class body can touch `Wallet`'s private fields and methods. (This isn't strictly true because `Wallet` might contain nested class definitions that don't automatically get access to `Wallet`'s private fields and methods, but aside from that, it's a useful rule of thumb.)

Submit

Show Answer

 Answers are displayed within the problem

## access control, part 6

1/1 point (graded)

Which of the following statements are true about the line marked `/*E*/` ? Check all that apply.

```
w.loanTo(w);
```

- ☒ The call to `loanTo()` is allowed by Java.
- ☐ The call to `loanTo()` is not allowed by Java because this and that will be aliases to the same object.
- ☐ The problem will be found by a static check.
- ☐ The problem will be found by a dynamic check.
- ☒ After this line, the `Wallet` object pointed to by `w` will have amount 0.
- ☐ After this line, the `Wallet` object pointed to by `w` will have amount 100.
- ☐ After this line, the `Wallet` object pointed to by `w` will have amount 200.



### Explanation

In this call to `loanTo()`, this and that will indeed be aliases for the same object, but Java doesn't prevent that. It causes `loanTo()` to behave badly, emptying out the wallet.

Submit

 Show Answer

 Answers are displayed within the problem

## access control, part 7

1/1 point (graded)

Which of the following statements are true about the line marked `/*F*/` ? Check all that apply.

```
return w.amount;
```

- ☐ The reference to `w.amount` is allowed by Java because both `w` and `amount` are private variables.
- ☐ The reference to `w.amount` is allowed by Java because `amount` is a primitive type, even though it's private.
- ☒ The reference to `w.amount` is not allowed by Java because `amount` is a private field in a different class.
- ☒ The illegal access is caught statically.
- ☐ The illegal access is caught dynamically.



Submit

 Show Answer

 Correct (1/1 point)

## access control, part 8

1/1 point (graded)

Which of the following statements are true about the line marked `/*G*/` ? Check all that apply.

```
return Wallet.amount == 0;
```

☐ The reference to `Wallet.amount` is allowed by Java because `Wallet` has permission to access its own private field `amount`.

☐ The reference to `Wallet.amount` is allowed by Java because `amount` is a static variable.

☒ The reference to `Wallet.amount` is not allowed by Java because `amount` is a private field.

☒ The reference to `Wallet.amount` is not allowed by Java because `amount` is an instance variable.

☒ The illegal access is caught statically.

☐ The illegal access is caught dynamically.



### Explanation

`amount` is an instance variable, so it requires a specific `Wallet` object instance to access. `amount` is also private, so not accessible here. Both problems would have to be fixed -- changing `amount` to public static -- in order to make this reference allowed.

Submit

Show Answer

Answers are displayed within the problem

Previous

Next

Some Rights Reserved

[Open Learning Library](#)

[About](#)

[Accessibility](#)

[All Courses](#)

[Why Support MIT Open Learning?](#)

[Help](#)

[Connect](#)

[Contact](#)

[Twitter](#)

[Facebook](#)

[Privacy Policy](#) [Terms of Service](#)

© Massachusetts Institute of Technology, 2024