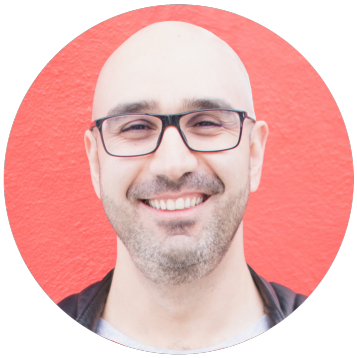
Classes

Quiz

Code with Mosh (codewithmosh.com)

1st Edition

About the Author



Hi! My name is Mosh Hamedani. I'm a software engineer with two decades of experience and I've taught over three million people how to code or how to become a professional software engineer. It's my mission to make software engineering simple and accessible to everyone.

https://codewithmosh.com

https://youtube.com/user/programmingwithmosh

https://twitter.com/moshhamedani

https://facebook.com/programmingwithmosh/

Questions

1- What is the difference between a class and an object?

2- What does instantiating mean?

3- What is the difference between stack and heap memory? How are they managed?

4- What are the problems of procedural code? How does object-oriented programming help solve these problems?

5- What is encapsulation?

6- Why should we declare fields as private?

7- What is abstraction?

8- What is coupling?

9- How does the abstraction principle help reduce coupling?

10- What are constructors?

11- What is method overloading?

12- What are static methods?

Answers

1- A class is a blueprint or template for creating objects. An object is an instance of a class.

2- Instantiating means creating an instance of a class: **new Customer()**

3- Stack is used for storing primitive types (numbers, boolean and character) and variables that store references to objects in the heap. Variables stored in the stack are immediately cleared when they go out of scope (eg when a method finishes execution). Objects stored in the heap get removed later on when they're no longer references. This is done by Java's garbage collector.

4- Big classes with several unrelated methods focusing on different concerns and responsibilities. These methods often have several parameters. You often see the same group of parameters repeated across these methods. All you see is procedures calling each other passing arguments around.

By applying object-oriented programming techniques, we extract these repetitive parameters and declare them as fields in our classes. Our classes will then encapsulate both the data and the operations on the data (methods). As a result, our methods will have fewer parameters and our code will be cleaner and more reusable.

5- Encapsulation is the first principle of object-oriented programming. It suggests that we should bundle the data and operations on the data inside a single unit (class).

6- How we store data in an object is considered an implementation detail. We may change how we store the data internally. Plus, we don't want our objects to go into a bad state (hold bad data). That's why we should declare fields as private and provide getters and or setters only if required. These setters can ensure our objects don't go into a bad state by validating the values that are passed to them.

7- Abstraction is the second principle of object-oriented programming. It suggests that we should reduce complexity by hiding the unnecessary implementation details. As a metaphor, think of the remote control of your TV. All the complexity inside the remote control is hidden from you. It's abstracted away. You just work with a simple interface to control your TV. We want our objects to be like our remote controls.

8- Coupling represents the level of dependency between software entities (eg classes). The more our classes are dependent on each other, the harder it is to change them. Changing one class may result in several cascading and breaking changes.

9- By hiding the implementation details, we prevent other classes from getting affected when we change these details. For example, if the logic board and transistors inside a remote control change from one model to another, we're not affected. We still use the same interface to work with our TV. Also, reducing these details and exposing fewer methods makes

our classes easier to use. For example, remote controls with fewer buttons are easier to use.

10- Constructors are called when we instantiate our class. We use them to initialize our objects. Initialization means putting an object into an early or initial state (eg giving it initial values).

11- Method overloading means declaring a method with the same name but with different signatures. The number, type and order of its parameters will be different.

12- Static methods are accessible via classes, not objects.