SWEN20003
Object Oriented Software Development

Classes and Objects 1 - Questions

Bach Le
bach.le@unimelb.edu.au

University of Melbourne

Learning Outcomes

Upon completion of this topic, which includes three lectures, you will be able to:

- Explain the difference between a *class* and an *object*
- Create classes, give them *properties* and *behaviours*, implement and use simple classes
- Identify a series of well-defined classes from a *specification*
- Understand the role of *getters*, *setters* and *constructors*
- Understand the differences between *instance*, *static* and *local* variables
- Understand the role of *standard methods* in java
- Explain object oriented concepts: *abstraction, encapsulation, information hiding and delegation*
- Understand the role of *wrapper* classes

Overview

This topic will be delivered through three lectures (Lectures 3, 4 and 5) each covering the following subtopics.

Classes and Objects - 1

- Introducing Classes and Objects
- Defining Classes
- Using Classes

Classes and Objects - 2

- Getters, Setters and Constructors
- Static Attributes and Methods
- Standard Methods in Java

Classes and Objects - 3

- Introducing Java Packages
- Information Hiding
- Delegation through Association
- Wrapper Classes

Which one of the following basic concepts is the main differentiator of the object oriented software design paradigm?

1. selection
2. output
3. iteration
4. calculation
5. none of the above

Answer:
(5) none of the above

abstraction - the way self-contained, reusable units are generated. In object oriented programming this is done through creating *classes* which have *attributes* and *methods.*

Which one of the following basic concepts is the main differentiator of the object oriented software design paradigm?

1. selection
2. output
3. iteration
4. calculation
5. none of the above

Answer:
(5) none of the above

abstraction - the way self-contained, reusable units are generated. In object oriented programming this is done through creating *classes* which have *attributes* and *methods*.

Describe the difference between the terms *class* and *objects*.

Answer:

Class

A definition or template for the behaviour and properties shared by a group of objects. For example, a `StuffedAnimal` defines what it means to be a stuffed animal (colour, size, weight).

Object

A particular set of data that fits its class' definition. For example, a `teddyBear` may be a `StuffedAnimal` that is *large*, *brown*, and *light*. Objects are sometimes also called *instances*, as they are a particular *instance* of a class.

A variable whose value is associated with an object of a class is called:

1. an instance variable
2. a local variable
3. a global variable
4. a class variable
5. none of the above

Answer:
(1) an instance variable

local variable - within a method instance variable - within a class, but the value is specific to the object (instance)

The body of a method that returns a value must contain at least one _________ statement.

1. void
2. invocation
3. throws
4. return
5. public

Answer:
(4) return

The body of a method that returns a value must contain at least one _________ statement.

1. void
2. invocation
3. throws
4. return
5. public

Answer:
(4) return

What is the output of the following program?

```java
public class DemoNoObject {
    public static void increment(int input) {
        input = input + 1;
    }
    public static void main(String[] args) {
        int a = 45;
        increment(a);
        System.out.println("a=" + a);
    }
}
```

Answer: a=45

What is the output of the following program?

```java
public class IntegerHolder {
    public int value;
    public void incrementValue(){
        value = value + 1;
    }
}
```

```java
public class DemoWithObject {
    public static void increment(IntegerHolder x) {
        x.incrementValue();
    }
    public static void main(String[] args) {
        IntegerHolder a = new IntegerHolder();
        a.value = 45;
        increment(a);
        System.out.println("a=" + a.value);
    }
}
```

Answer: a=46

Define the object oriented terms, Data Abstraction and Encapsulation.

Answer:

Keyword

Data Abstraction: The technique of creating new data types that are well suited to an application by defining new classes.

Keyword

Encapsulation: The ability to group data (attributes) and methods that manipulate the data to a single entity though defining a class.

The new operator:

1. Allocates memory
2. Is used to create an object of a class
3. Associates an object with a variable that names it
4. All of the above
5. None of the above

Answer:
(4) All of the above

The new operator:

1. Allocates memory
2. Is used to create an object of a class
3. Associates an object with a variable that names it
4. All of the above
5. None of the above

Answer:
(4) All of the above

What does ‘null’ mean in Java?

Answer:
`null` is a special value typically used to signify:

- An unknown value
- An object that doesn’t exist
- A state that has yet to be initalised

Develop a system (a set of classes) that can "replicate" the behaviour of Internet Movie Database (IMDB). It should be able to store the details of actors, movies, their associations with each other, and their ratings.

How would you go about developing this? What are your classes? What are the attributes and methods for each class?

IMDB Clone

What classes can we use for our example problem?

Fundamental:

- `Actor`
- `Movie`
- `Database` ("main")

Additional:

- `Rating`
- `Comment`

IMDB Clone

What attributes and methods can we add to our classes?

`Actor`:

- Attributes
  - name
  - age
  - country
  - appearances
  - rating
- Methods
  - print
  - appearsIn

IMDB Clone

What attributes and methods can we add to our classes?

`Movie`:

- Attributes
  - title
  - earnings
  - actors
  - rating
- Methods
  - print
  - hasActor

IMDB Clone

What attributes and methods can we add to our classes?

`Database`:

- Attributes
  - actors
  - movies
- Methods
  - main
  - createActors
  - createMovies
  - search

Learning Outcomes:

Topics covered in this lecture:

- Introducing Classes and Objects
- Defining Classes
- Using Classes