Classes and Objects

Object Orientated Analysis and Design

Benjamin Kenwright

Outline

- Review Previous Weeks
  - Object Model, Complexity, ..
- What do we mean by Classes and Objects?
- Summary/Discussion

Review

- Last Week Object Model and Evolution of Software Engineering
- Familiar with Chapters 1, 2 and 3
  - Read Chapter 3 Last Week
- Reviewing material regularly
  - e.g., complexity, object orientated concepts, ..

Revision Question

- Write down the four elements that Inherent Complexity derives from?

  (5 Minutes)

Answer

- Inherent Complexity derives from four elements:

1. complexity of the problem domain
2. difficulty of managing the development process
3. flexibility possible through software
4. problems of characterizing the behavior of discrete systems

Revision Question

- What are the three important parts of Object-Oriented Programming (OOP)?

a) uses objects; each object is an instance of some class; classes may be related to one another via inheritance

b) use modules; hierarchical structure; structures must be related to one another via inheritance

c) hierarchical structure; collection of objects; objects must be related to one another via polymorphism

Answer

- a) uses objects; each object is an instance of some class; classes may be related to one another via inheritance

Revision Question

- Why is modularity important?

a) Enables us to partitioning a program into individual components can reduce its complexity
b) Enables us to develop more optimised algorithms
c) Modularity causes issues with boundaries (or interfaces) within the program

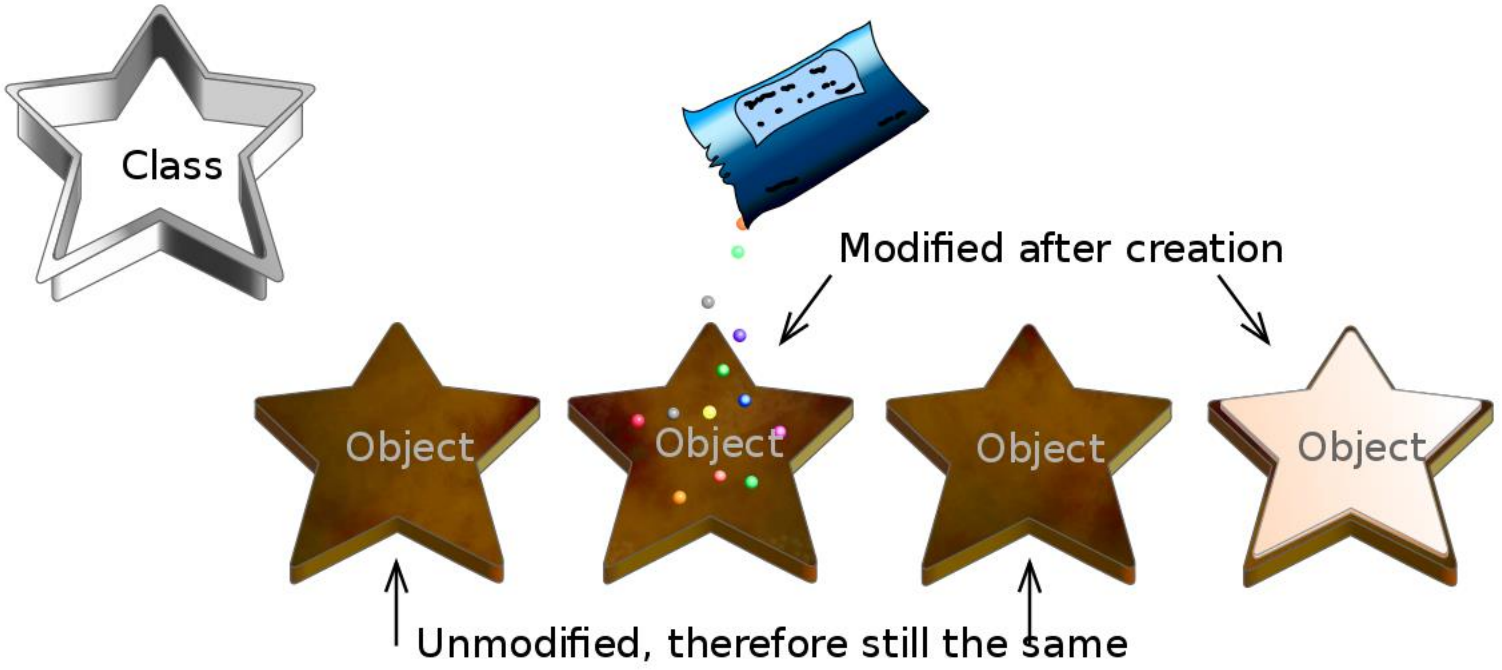
Answer

- a) Enables us to partitioning a program into individual components can reduce its complexity

Building Blocks

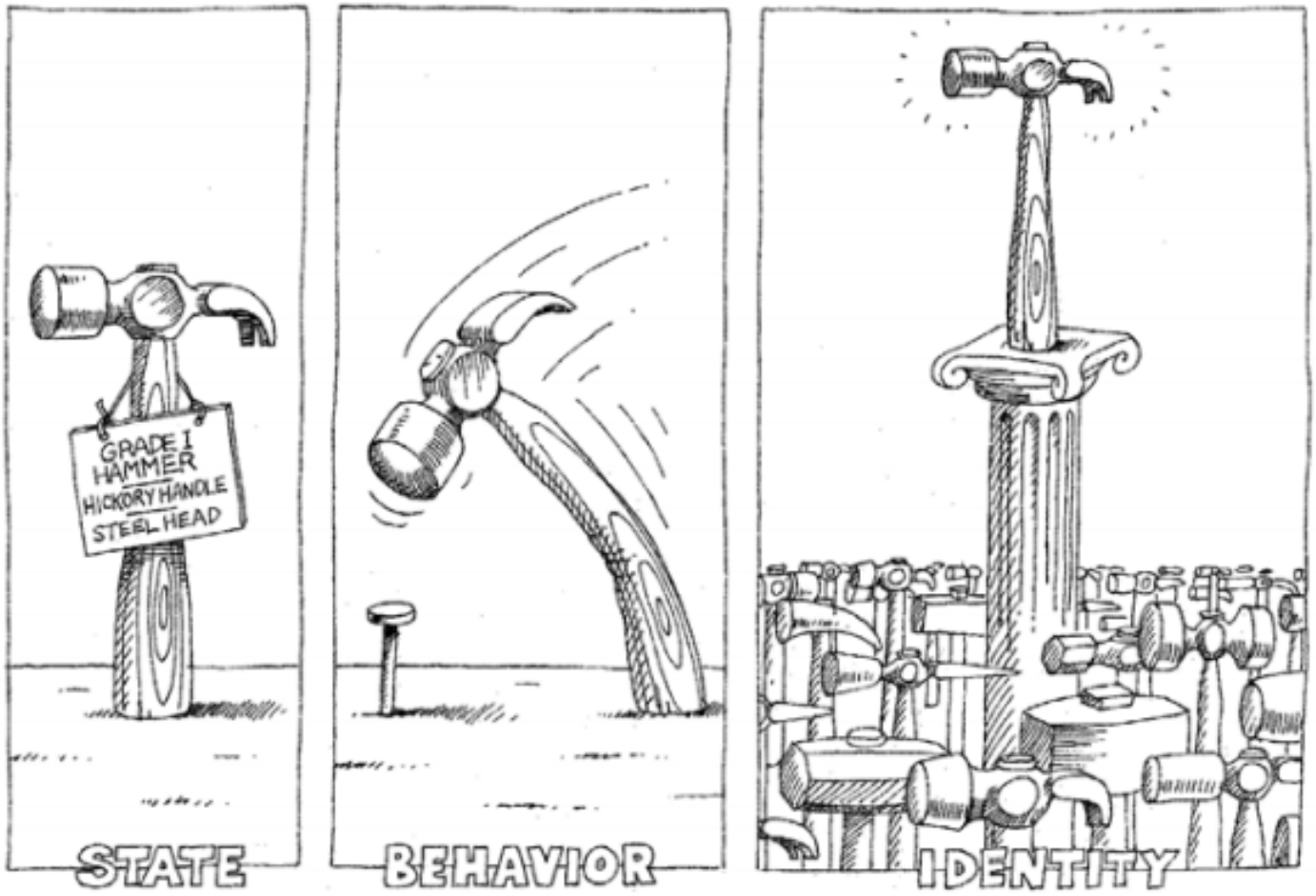
- When we use object-oriented methods to analyze or design a complex software system, our basic building blocks are classes and objects

What Is and What Isn't an Object?



What Is and What Isn't an Object?

- An object is an entity that has state, behavior, and identity. The structure and behavior of similar objects are defined in their common class. The terms instance and object are interchangeable



An object has state, exhibits some well-defined behavior,
and has a unique identity.

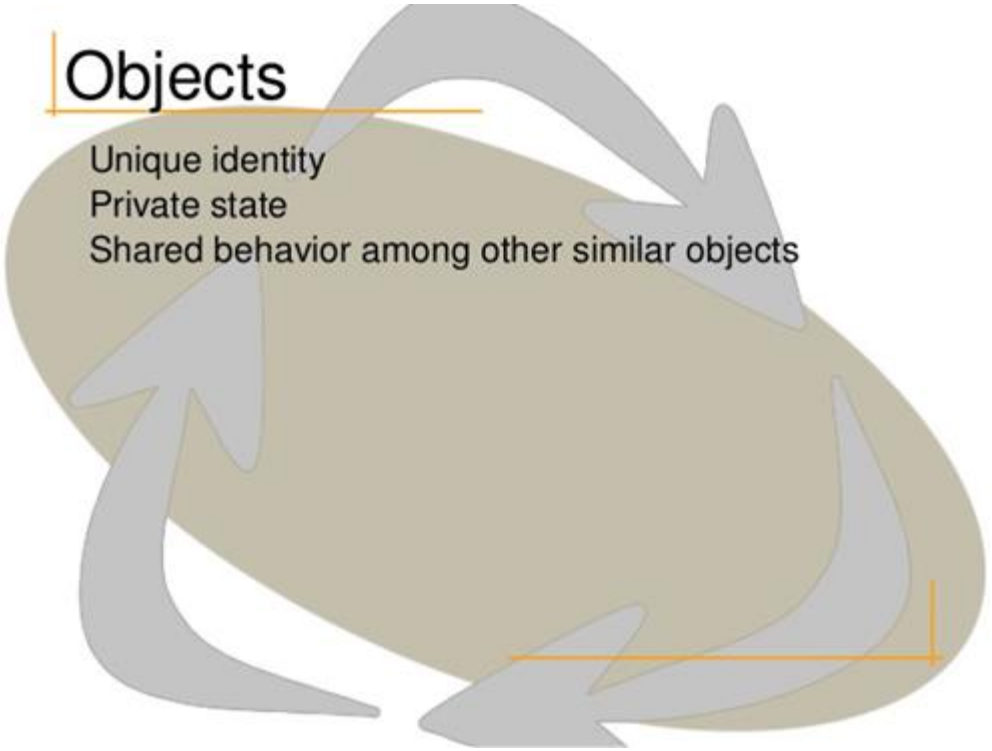
Question

- An object is an entity that has:

a) state, action, dependencies
b) identity, state, behavior
c) behaviour, action, state
d) class, state, memory

Answer

- b) identity, state, behavior



State

- The state of an object encompasses all of the (usually static) properties of the object plus the current (usually dynamic) values of each of these properties

- Examples of State Properties
  - ▷Elevators travel up or down
    - Current floor
  - ▷Vending machines accept coins
    - Number of coins deposited
  - ▷Clocks indicate the current time
    - The number of minutes since the last hour

Behavior

- Behavior is how an object acts and reacts, in terms of its state changes and message passing

- Example: Behavior of a car: braking, changing gears, opening doors, moving forwards or backwards, ..

Identity

- Identity is that property of an object which distinguishes it from all other objects

Question

- State is how an object acts and reacts, in terms of its state changes and message passing

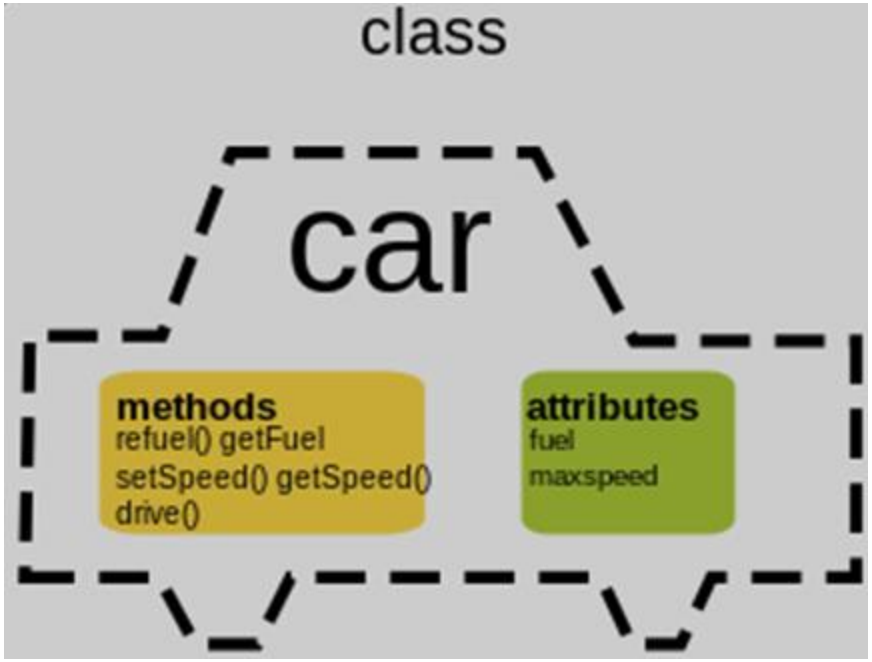
a) True
b) False

Answer

- b) False

Behavior is how an object acts and reacts, in terms of its state changes and message passing.

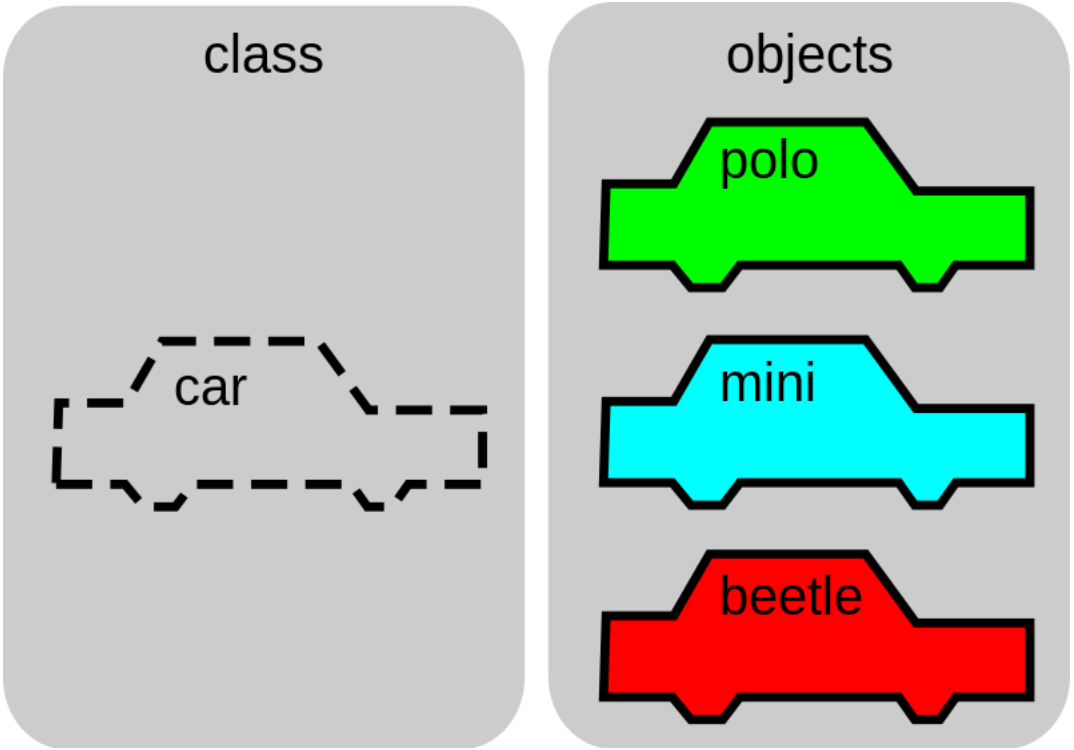
What Is and What Isn't a Class?



What Is and What Isn't a Class?

- A class is a set of objects that share a common structure, common behavior, and common semantics
- A single object is simply an instance of a class
- What isn't a class?
  - An object is not a class. Objects that share no common structure and behavior cannot be grouped in a class because, by definition, they are unrelated except by their general nature as objects

Review



Inheritance

Inheritance

- Inheritance is an interesting concept for representing concrete relationships
  - Including generalization/specialization relationships
- A subclass may inherit the structure and behavior of its superclass

Polymorphism

- Generally, the ability to appear in many forms. In object-oriented programming, polymorphism refers to a programming language's ability to process objects differently depending on their data type or class. More specifically, it is the ability to redefine methods for derived classes
- Helps develop more concise clean error free code
  - Without polymorphism, the developer ends up writing code consisting of large case or switch statements (conditional logic)
- Inheritance without polymorphism is possible, but it is certainly not very useful
- Polymorphism and late binding go hand in hand

The Role of Classes and Objects in Analysis and Design

- During analysis and the early stages of design, the developer has two primary tasks:

1. Identify the classes that form the vocabulary of the problem domain

2. Invent the structures whereby sets of objects work together to provide the behaviors that satisfy the requirements of the problem

Measuring the Quality of an Abstraction

- How can one know if a given class or object is well designed?

Measuring the Quality of an Abstraction

- Five meaningful metrics to measure the quality of abstraction are:

1. Coupling
2. Cohesion
3. Sufficiency
4. Completeness
5. Primitivenes

Question

- What are the five meaningful metrics to measure the quality of abstraction?

a) Contracts, Cohesion, Completeness, Primitivenes, Sufficiency

b) Cohesion, Coupling, Sufficiency, Completeness, Primitivenes

c) Coupling , Cohesion, Safety, Completeness, Primitivenes

Answer

- b) Cohesion, Coupling, Sufficiency, Completeness, Primitivenes

Interfaces and Implementations

- Larger problem is decomposed into smaller problems (e.g., classes)
- Binding contract between classes
- Interface of a class provides information on what is viewable to the outside (contracted rules)
- Interface allows a class to become more formal about the behavior it promises to provide

Interface

- The interface of a class is divided into four accessibility levels:

1. Public: a declaration that is accessible to all clients
2. Protected: a declaration that is accessible only to the class itself and its subclasses
3. Private: a declaration that is accessible only to the class itself
4. Package: a declaration that is accessible only by classes in the same package

Advantages of Interfaces

- Interfaces solve many problems associated with code reuse in object-oriented programming
- Allows the construction of flexible dependencies for a class definition
- Dependencies make it easier to maintain or extend the class without breaking the client
- Resolves the problem of having tedious or impossible to improve the code
- Helps with maintainability and extensibility

Question

- Four accessibility levels of an interfaces are:

a) public, protected, private, proactive

b) protected, public, package, passive

c) public, private, passive, proactive

d) package, public, protected, private

Answer

d) package, public, protected, private

Relationships

- The three primary kinds of relationship between components are:

1. Association
2. Inheritance and
3. Aggregation

Association

- A relationship denoting a semantic connection between two classes

Aggregation

- A whole/part relationship where one object is composed of one or more other objects, each of which is considered a *part of the whole*. This relationship is a weak form of containment in that the lifetimes of the whole and its parts are independent

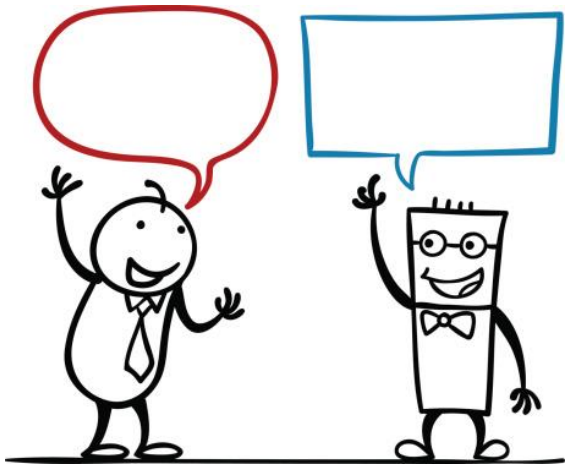
Inheritance

- A relationship among classes, wherein one class shares the structure or behavior defined in one (single inheritance) or more (multiple inheritance) other classes. Inheritance defines an “is a” hierarchy among classes in which a subclass inherits from one or more generalized superclasses; a subclass typically specializes its superclasses by augmenting or redefining existing structure and behavior

Discussion Activity

- Explain the differences between Aggregation and Inheritance?

  (5 Minutes)



Aggregation vs Inheritance

- Aggregation: create new functionality by taking other classes and combining them into a new class. Attach an common interface to this new class for interoperability with other code

- Inheritance: extend the functionality of a class by creating a subclass. Override superclass members in the subclasses to provide new functionality. Make methods abstract/virtual to force subclasses to "fill-in-the-blanks" when the superclass wants a particular interface but is agnostic about its implementation.

Design Decisions

- Criteria to be considered when making class design decisions:
  - Reusability: Would this behavior be more useful in more than one context?
  - Complexity: How difficult is it to implement the behavior?
  - Applicability: How relevant is the behavior to the type in which it might be placed?
  - Implementation knowledge: Does the behavior's implementation depend on the internal details of a type?

Question

- What are four criteria considered when making class design decisions:

a) reusability, complexity, implementation knowledge, attributes

b) complexity, reusability, applicability, implementation knowledge

c) complexity, scalability, applicability, implementation knowledge

d) performance, scalability, applicability, implementation knowledge

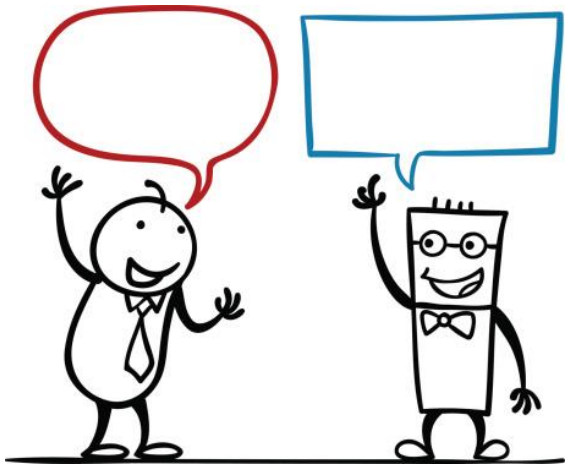
Answer

- b) complexity, reusability, applicability, implementation knowledge

Discussion Activity

- Explain some of the trade-offs decisions that might be made during the analysis and design of a software project?

  (5 Minutes)



What is the Law of Demeter?

What is the Law of Demeter?

- The Law of Demeter (LoD) or ‘principle of least knowledge’ is a design guideline for developing software, particularly object-oriented programs
- "Only talk to your friends" is the motto
- Reducing the dependencies between classes to make your code more flexible (i.e., reduces coupling between objects)
  - ▷`low-coupling’
- Each component should have only limited knowledge about other units

Advantages of LoD

- The advantage of following the Law of Demeter is that the resulting software tends to be more maintainable and adaptable
- Since objects are less dependent on the internal structure of other objects, object containers can be changed without reworking their callers

Disadvantages of LoD

- May also result in having to write many wrapper methods to propagate calls to components; in some cases, this can add noticeable time and space overhead
- Trade-off Solution – Not always positive
  - Augment a number of methods into modules

Summary

- Clear understanding of classes and objects
- An object has state, behavior, and identity
- The structure and behavior of similar objects are defined in their common class
- Behavior is how an object acts and reacts in terms of its state changes and message passing

This Week

- Review Slides
- Quizzes Online
- Read Chapter 4
- Project
  - Groups (2-3 People)
  - Create Version Control Repository
    - GitHub
    - Submit Repository Name

Group Project

- **Design ATM System**
  - i.e., Automated Teller Machine (Cash Machine)
- Research & Investigation
  - Read around the topic
- Features/Operations
  - Custom Details
  - Bank Database
  - Security/Validation
  - Flow Diagrams/Operations/States

Project

- Start Early!
- Time Management
  - Evidenced (e.g., Version Control)
  - Each person must submit report/details/logs
  - Each team should have a `custom' solution

Questions/Discussion

Question

- Identity is that property of an object which distinguishes it from all other objects

a) True

b) False

Answer

- a) True

Question

- Inheritance create new functionality by taking other classes and combining them into a new class. Attaching common interfaces to this new class for interoperability with other code

a) True

b) False

Answer

b) False

- Aggregation: create new functionality by taking other classes and combining them into a new class. Attach an common interface to this new class for interoperability with other code

Question

- A class is the same as an object. Objects share common structure and behavior as a class because and by definition are related by their general nature as objects

a) True

b) False

Answer

- b) False

An object is not a class. Objects that share no common structure and behavior cannot be grouped in a class because, by definition, they are unrelated except by their general nature as objects

Question

- An object-oriented program consists of many objects. Each object has the same identity, state (attributes, data, and their current values) and behavior (operations)

a) True

b) False

Answer

- b) False

An object-oriented program consists of many objects. Each object has the same (wrong) identity, state (attributes, data, and their current values) and behavior (operations)

Classes and Objects are the Building Blocks for Analysis and Design Solutions for Complex Systems

Individual Have Different Perspectives/Views – Clear Set of Requirements/Specifications