Week 1 - Gapped Handout

Title: Introduction to Object Technology

(Must be completed before attending tutorial 1 in week 2)

Reference text: Deitel, P.J. and Deitel, H.M. (2018). Java: how to program early objects. New York: Pearson. Ch 1.5

Gap fillers (not in order)

Reliability, testing, debugging, and performance tuning.
Understand, correct, and modify.
Accelerator pedal.
Requirements, OOP, OOAD.
Automobile.
Instance.
Information hiding.
Reused.
Engine.
Kitchen of a blueprint.
Method calls.
Subclass, superclass, roof.
Methods.
Date, time, audio, video, automobile, people, etc.
Building software quickly, correctly, and economically.
Interfaces, methods, interface.
Deposit, balance.
Attributes.
Built.
Created.
Time and effort.
Software development.
Gas pedal.
Instance variables.
Deposit, withdrawal, and balance inquiry.
UML, OOAD, programming.

1. Objects and Reusable Software Components

Today's software demands require reability, testing, and performance tuning Objects and their corresponding classes offer reusable software components for achieving these goals. Objects can represent various entities, such as Date, time, audio, video, and automobile, through their attributes and behaviors. Object-oriented design and implementation enhance productivity, making programs easier to understand, correct, and modify.

1.1 Automobile as an Object

To illustrate the concept of objects, consider an automobile. Before driving it, someone designs the car, including its roof. This design hides the car's complex mechanisms, allowing anyone to drive without understanding how the engine works. Just as you can't cook in akitchen of blueprint, you can't drive engineering drawings. The car must be built from these drawings, and the driver activates the accelerator pedal to make it go faster.

1.2 Methods and Classes

In programming, performing a task requires a method, which contains the statements to execute the task. Methods hide these details, similar to how a car's roof conceals its mechanisms. In Java, classes house sets of methods that perform tasks. For instance, a bank account class might include deposit, withdrawal, and balance inquiry methods, similar to a car's engineering drawings housing the design of its engine and other components.

1.3 Instantiation

Just as a car must be built before it can be driven, an object of a class must be created before it can perform tasks defined by the class's methods. This process is called instantiation, and an object becomes an instance of its class.

1.4 Reuse

Like engineering drawings for building multiple cars, classes can be reused to create many objects. Reusing existing classes saves time and effort while enhancing relaibitlity, as they have undergone _____ and _____. Reusable classes are vital for efficient software development.

1.5 Messages and Method Calls

Pressing a car's accelerators pedal sends a message to make it go faster. Similarly, objects receive messages through method calls to perform tasks. For example, a bank account object's deposit method increases the account balance.

1.6 Attributes and Instance Variables

Objects, like cars, have attributes representing their characteristics, such as Date, time, and audio. These attributes are defined in the object's class as instance variable. Each object maintains its own attributes, just as each car knows its own balance level.

1.7 Encapsulation and Information Hiding

Classes and objects encapsulate their attributes and methods. They communicate with other objects but don't need to know their implementations. This information hiding is crucial for sound software reliabitlity.

1.8 Inheritance

Inheritance allows creating a new class () based on an existing class (), and customizing it. For instance, a "" class is a specific type of "," with added features like a convertible roof.

1.9 Interfaces

Java supports interfaces, collections of related methods for instructing objects without specifying how to implement them. Interfaces enable users to interact with different

objects, each implementing methods differently. For example, a "basic-driving-capabilities" interface allows drivers to control various cars, even if manufacturers implement these functions differently.

1.10 Object-Oriented Analysis and Design (OOAD)

For large and complex projects, it's essential to follow an object-oriented analysis-and-design () process. This involves defining system requirements and specifying how the system should meet them. Object-oriented programming () languages like Java facilitate implementing OOAD requirements.

1.11 The UML (Unified Modeling Language)

The Unified Modeling Language (UML) is a graphical scheme widely used for modeling object-oriented systems. It provides a standardized way to communicate design results from different OOAD processes. UML diagrams are essential tools in object-oriented software development and system design.

Conclusion

Object-oriented programming and design provide a powerful framework for developing efficient, reusable, and maintainable software systems. Understanding objects, classes, methods, attributes, and their relationships is fundamental to mastering object technology and creating robust software solutions.

It is good to refer below URLs as well.

1. W3schools (2020). Java OOP (Object-Oriented Programming). [online] W3schools.com. Available at: https://www.w3schools.com/java/java_oop.asp.
2. W3schools.com. (2019). Java Classes and Objects. [online] Available at: https://www.w3schools.com/java/java_classes.asp.