RKR

Introduction to Object Oriented Paradigm Concepts

Ringa K.R

RKR

Lecture objectives

- To review the basic principles of Object Orientation.

RKR

Lecture Overview

We will cover the following principles as we do the review of principles of Object Orientation:

- objects
- classes
- encapsulation
- inheritance
- polymorphism

RKR

Object Oriented System

"Object-oriented system is composed of objects. The behaviour of the system is achieved through collaboration between these objects, and the state of the system is the combined state of all the objects in it."

RKR

Object defined

Tangible or visible thing; or

Something to which thought or action is directed;

In either case, it is something that may be apprehended for analysis.

RKR

Object

An object is:
"An abstraction of something in a problem domain, reflecting the capabilities of a system to keep information about it, interact with it, or both" (Coad & Yourdon)

"An entity able to save state *(information)* which offers a number of operations (*behaviour*) to either examine or affect this state" (Jacobson)

"An individual unit of *run-time* data storage that is used as the basic building block of programs" wikipedia

RKR

Object

- Object: highly cohesive, self managing entity with the following characteristics:
  - Attributes;
  - Behaviour;
  - Identity (Unique identifier).
- Additionally, the combination of attributes and behaviour from instantiation of object give an object **state.**
- The object's behaviour gives defined set of **responsibilities.**
- Together, we may regard these as an **object's qualities.**

RKR

Object Interaction

For objects to interact, they have to be activated or stimulated. This is achieved by passing messages between objects.

The receipt of a message activates the object and the object will execute an operation that is identified by the message.

Operations are, therefore named and this name is usually used in the message. The message may also include parameters that the operation will need for execution. Once the operation is finished the result is sent back to the sender of the first message.

RKR

Classes

A class represents a group of objects which have the same behaviour and information structures.

"A class represents a template for several objects … objects of the same class have the same definition both for their operations and for their information structure" (Jacobsen)

Class is a kind of type, an ADT (but with data), or an 'entity' (but with methods).

Classes are the same in both analysis and design.

RKR

Classes

There are three major components that define a class:

- we define the attributes it has,
- the relationship instances that the class can participate in, and
- the operations that instances of the class can execute.

RKR

Encapsulation

Encapsulation means, literally, turning something into a capsule, i.e. into a small, sealed container whose contents are normally invisible from the outside, with consequences for 'visibility' of both operations and data.

"Behaviour and information are encapsulated in objects" (Jacobsen)

"Encapsulation refers to the bundling of data with the methods that operate on that data."

RKR

Encapsulation

Encapsulation: how data and operations (the properties of an object) are bound together in classes:

- Provides measures of privacy of data within a class (data-hiding);
- Makes objects self-managing instantiations of their classes;
- Employs relationships of aggregation, composition, dependency and association.

RKR

Encapsulation

Encapsulation in relation to software objects, is concerned with:

- only the 'interface' of an object that is 'visible' to other objects
- need, and can, only know operations on an object, not how they work nor about other characteristics
- necessary prerequisite for information hiding.

RKR

Inheritance

In object-oriented programming, **inheritance** is a way to form new classes (instances of which are called objects) using classes that have already been defined.

The new classes, known as **derived classes**, take over (or **inherit**) attributes and behaviour of the pre-existing classes, which are referred to as **base classes** (or ancestor classes).

It is intended to help reuse of existing code with little or no modification.

RKR

Inheritance

We can identify more abstract classes that specify a number of attributes, relationships and operations. Then we can specify other classes to inherit these more abstract classes. The more abstract class is called super – class and the more concrete class is called sub – class. A sub – class inherits all properties from its super – class.

RKR

Inheritance

Inheritance: concept of "write once, test until satisfied once, use many times":

- How properties of one (base) class may, selectively, be incorporated into other (derived) classes for re-use;
- Employs relationship of generalisation.

Two important analytic activities, 'generalization' and 'specialization', are associated with the principle of inheritance.

RKR

Inheritance

- Generalization means extracting common properties from a collection of classes and placing them higher in the inheritance hierarchy, in a 'super class'.
- Specialization involves the definition of a new class which inherits all the characteristics of a higher class and adds some new ones, in a 'sub class'.
- Specialization is a top-down activity that refines the abstract class into more concrete classes and generalization is a bottom – up activity that abstracts certain principles from existing classes in order to find more abstract classes.

RKR

Polymorphism

In simple terms, **polymorphism** lets you treat derived class members just like their parent class's members.

In more precise terms, polymorphism (object-oriented programming) is the ability of objects belonging to different types to respond to method calls of methods with the same name, each one according to an appropriate type-specific behaviour.

The programmer (and the program) does not have to know the exact type of the object in advance, so this behavior can be implemented at run time (this is called *late binding* or *dynamic binding*).

RKR

Polymorphism

Polymorphism: concept of one interface for differing applications:

- Limited polymorphism achieved via overloading of operations (static polymorphism);
  - Compiler resolves which function to employ at compile-time, discriminating via operation signature;
- True polymorphism achieved via inheritance mechanism (run-time polymorphism):
  - Objects accessed identified by generic identifier (of base class) and run-time program executable utilizes class definition to choose particular class operation on data.