**Object-oriented programming concepts**

An Introduction

---

**Programs & Objects**

- **Program** consists of a set of interacting objects
- **Object** comprised of data & its associated operations; represents some tangible or intangible concept in program’s problem domain
- **Examples:**
  - Bank account
  - Robot
  - Song

---

**Classes**

- **Class**: defines characteristics of an object to be created
  - Dictates what object can & cannot do
  - Object is instance of class
  - Can create many instances of a class
- To accomplish tasks in a program, we tap into capabilities and characteristics of objects & classes
- Classes and objects have two different kinds of **members**: methods and data

---
Messages and Methods

- **Message**: a single program instruction that requests a particular task to be performed; *calls on* a class or object to execute a method
- **Method**: the set of instructions that will be performed in response to a message; the implementation of an algorithm

Arguments

- **Argument**: a data value (*expression*) we pass along with a message
  - Supplies the method with the raw material we need it to process
  - Examples:
    - amount of money to set for minimum balance
    - amount to be withdrawn or deposited
    - distance for robot to travel
    - angle of turn
    - volume at which to play song

Data members

- **Data members**: information held/manipulated by an object or class
- Class definition lists the types of data values each of its instances can contain; can be:
  - **Constant**: a data value that is set at its definition and cannot be changed during the run of a program (example: the pitch frequency of a particular musical note)
  - **Variable**: a data value that may or may not be set at definition, but which may be changed (account balance, robot speed)
**Data members**

- **Class data value**: single value shared by all instances of class (minimum balance, maximum speed)
- **Instance data value**: unique value held by particular object (my account balance, your robot’s current location, the length of a song)

**Inheritance**

- **Inheritance**: mechanism that allows classes that share common characteristics to be defined as extensions of another class
- **Superclass**: class that contains definitions of common characteristics; also called parent class
- **Subclass**: class that is derived from a superclass; contains specialized characteristics; also called child class

**Inheritance**

- Child classes inherit common characteristics from parent classes, but differ from each other in some specialized way; examples:
  - All bank accounts have a balance, but a savings account may pay interest whereas a checking account may not
  - All songs contain a set of notes, but we can differentiate purely instrumental songs from songs with lyrics
- **An inheritance hierarchy** can be created in which a subclass becomes a superclass for other subclasses
Software life cycle

- **Software life cycle**: sequence of stages from conception to working program (and beyond); stages include:
  - Analysis
  - Design
  - Coding
  - Operation/Maintenance

Software life cycle

- **Analysis**: feasibility study to see if problem is solvable; result is requirements specification that describes program features in a verifiable form
- **Design**: turn requirements specification into a set of classes that fulfill the requirements
- **Coding**: implement design by writing a program (in Java or some other programming language) – a.k.a. **implementation phase**

Sub-stages of implementation

- **Testing**: run program with sample data to verify its compliance with specification (correctness)
- **Debugging**: elimination of programming errors found through program testing
- **Types of errors**:
  - **Syntax**: prevent program from compiling; violation of programming language rules
  - **Semantic**: program compiles and runs, but produces incorrect results
**Operation/Maintenance Phase**

- In the real world, this is the longest phase.
- Program is put into production; changes may be required over time.
- An important goal of programming is to write code that not only works, but is easily maintained.