Java Programming

Developing Solutions

Course Overview

- The purpose of this presentation is to present the basic mindset of programming in Java
  - The objective is to provide solutions
  - The prerequisites are Goals, planning, preparation, and measurement
  - The concepts of components, their properties, and behaviors are related to Java
- The sequence of the presentation relates a couple common scenarios to the perspective of programming
- A simple control model example is presented to demonstrate how analysis may result in multiple solutions, and the solutions may or may not require programming
- The presentation is limited to Java data types, class encapsulation, and packages

Introduction

- When we think of programming, we often think of people sitting at their keyboards writing the programs that solve problems, make the programmers rich, let us text each other in class.   Actually, when we use their software, call each other, use the GPS to navigate, watch YouTube videos, or perhaps play Pokemon Go, we seldom think about the programmers that make it possible.
- Those most successful in programming are those that solve problems for others. With each programmer making their contribution, large and small, collectively they have made the world goods more accessible and made tremendous gains in productivity, safety, and efficiency.
- Successful programmers have something in common. That is a “goal”.

Goals Setting / Objectives

- Goals give the programmer a focus and a basis for a solution. Specifically, if the program has no purpose, it is not likely to provide any benefit. When I refer to “solution”, I am referring to a program and associated systems that satisfy a purpose.
- Purpose
- Tasks
- Analysis
- Implementation
- Assessment/Feedback

Purpose

- A purpose provides a theme or context. Whether we are building a website to dole out videos or sell products, building robotic aerial vehicles to inspect the drilling tower of offshore oil rigs, we focus on our purpose to understand each task we are expected to perform.

Tasks

- Tasks are the sequence of activities we perform in the course of providing a solution.

Requirements

- Requirements specify the criteria we must satisfy. Whether it is returning a web page in under 2 seconds, performing very accurate calculations, climbing a pyramid, or throwing a ball through a hole in the wall, our solutions must meet these requirements, or fail.

Analysis

- Analysis is the task of understanding our goals, breaking them down into tasks, defining the requirements to be met, and coming up with a design that will put it all together to work together to meet our original goal. Analysis may come from the innovation of the team, or may be patterned from the work of others (as found on the Internet).

Implementation

- Implementation applies the approach and insight from the analysis into the mechanical and software systems that ultimately provide the benefits that meets our goal.

Assessment

- Assessment is how we determine of we met our objectives. Most often, we demonstrate we satisfied the requirements of the original goal. As you grow with providing solutions and programming, you will learn “success” is a minimum requirement. Other measures for cost, response time, reliability, and insight to leverage your solutions to other problems will become important.

Management

- Design Overview
- Verify Purpose, Requirements, Tasks
- Determine Task Implementation Sequence
- Assign Roles/Skills
- Monitor Activity
- Incremental Testing & Feedback
- Repeat until requirements met

Simple Tasks

- Things you frequently do are simple tasks. At least they seem simple to you because you have done them before.
- For example, when you are hungry, you open the refrigerator or pantry to see what it there that will satisfy your hunger. It is “analysis” as you select your candidates for a solution.
- When you decide on cereal, you go through the sequence of getting the cereal, milk, bowl, and spoon and the tasks of getting an appropriate mix of cereal and milk in the bowl. Hopefully, you include the tasks to clean up to put the remaining milk and cereal away.

Cookies (Exercise)

- The purpose of this exercise is to treat the making of cookies as a task. In performing this exercise, the focus will be on the sequence of steps required. The outcome should produce some very basic questions as well as some activities for preparation, process loop, parallel activities, time critical events, determination of finished, and cleanup. An assessment of the task is expected, including an awareness of duration, labor effort, and build-or-buy decisions.

Cookies (Exercise) (cont’d)

- Ingredients
- Tools/Equipment required
- Task sequence
- Setup
- Process Loop
- Cleanup

Complex Tasks

- Complex tasks are similar to simple tasks in that they solve a particular problem or provide a specific solution. Whereas complex tasks may also have a sequence of activities, they leverage other simple tasks and complex tasks to work together to provide a single solution.
- For example, taking a trip in a car.
  - Load, fill tank, plan route, navigate, arrive, unload

Turkey Dinner (Complex Example)

- The purpose of this exercise is to treat the making of a complex meal as a task.
- Similar to the simple tasks, the outcome should produce some very basic questions as well as some activities for preparation, process loop, parallel activities, time critical events, determination of finished, and cleanup.
- Unlike a simple task, the complex task may have tasks running in parallel and/or have tasks that require another task to finish first, or perhaps tasks that are paired and performed serially.
- An assessment of the task is expected, including an awareness of duration, labor effort, build-or-buy decisions, and whether or not it all came together as planned.

Turkey Dinner (Complex Example)

- **Requirement** – everything to be ready at the same time, …
- **Turkey** – roasted, cooled, and carved
- **Gravy** – turkey gravy piping hot
- **Potatoes** – mashed
- **Salad**
- **Bread**
- **Pumpkin Pie** – fresh, but cooled, with whipped cream
- **Ingredients**
- **Tools/Equipment required**
- **Task sequence**
- **Setup**
- **Process Loop**
- **Cleanup**

Task Analysis

- In analyzing tasks, a clear understanding of the measurable objective of the successful task and the requirements influencing the success is needed. When there are fundamental problems that detract from success, then reviewing different types of solutions and historical solutions is warranted. It may be solutions are ruled out because of the cost, complexity, or they are detrimental to the overall goal. It may also be that the simple, mechanical or manual solutions may be perfectly adequate and that extensive effort is not required.

Rocket Launch (Walk through)

- When a large rocket is launched, it attempt to rise vertically solely from the thrust of the solid or liquid fueled rocket motors.
- It is very unstable because the center of gravity is ahead of the thrust, and physics has the GC trying to get aft of the thrust.
- The fins on the rocket are not affective until it is flying sufficiently fast to get enough airflow form control.
- Control requirements change during use

Rocket Launch

- Rocket is unstable until it is up to speed.
  - How does a bottle rocket solve this problem?
  - How does a model rocket solve this problem?
  - Which one moves the center of gravity?

Rocket Launch – Controlled

- Gimbled Engine
- Inertial Vertical Guidance System – what do we know
- Controlling the gimbled engine – what do we know
- What if things change because of speed? add a controller

Draw a Picture

- Draw a picture of what you are trying to do.
- Easier to see what you need
- Easier for others to understand

- In activity development, we call these pictures a “Sequence Diagram” or a “Flow Chart”.

- In designing the objects we manage, we call them “Object Model Diagrams”. (“OMD”)

Object/Encapsulation

- What we need to remember?
- What we know when we use it
- What can we make it do?
-

  Donut Diagram,
  - variable encapsulation,
  - Methods,
  - Maintaining integrity of the data

Components and Tasks/Packages

Management of reusable components and functions allows us to reuse major sections of our solutions, or bring in implementations found elsewhere.

- **By component**
- **By function**
- By vendor

Java

- Java is one of the six targeted programming languages for supportable development.
- It competes with:
  - Javascript
  - Python
  - Ruby
  - PHP
  - Go
- These languages allow for faster development than older languages, and support a very broad range

Why Java

- Platform independent
- Broad skill base (many people program)
- Widely adopted - Used in cell phones, web site, banking, Facebook, search engines
- Many real-world examples on the Internet
  Supports Unicode (multiple languages)
  Memory Management is suitable for long running applications
- Excellent network capability
- Very enhanced with packages where working
- components are usually available
- Supports many frameworks (most of the programming is done for you, and you add your part)

JDK

- The **Java Development Kit** (JDK) is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc) and other tools needed in Java development.
- In practice, most developers use an IDE to develop and deploy (install) their java applications.

Portable

- Java was developed to run inside a Java Runtime Environment (JRE).
- The JRE handles the connections to the system (computer) resources. Using the JRE, all machine appear to be equal.
- The adage is, “Java is a language you can write once and run anywhere.”
- Not only does this mean your skills will work on many platforms, it means your code will also work on many platforms without having to write machine-specific code.

Java Class – Donut Diagram

- Variables
  - Class variables
  - Local variables
- Data Types
  - Numbers
  - Boolean
  - String
  - References
- Methods
- Classes (Defined Java Class Files)

Classes – Data Centric

- Data Centric classes are used primarily for holding data.

Classes – Behavior Centric

- Behavior centric classes primarily contain the methods we use for particular purposes:
    - Formatting output and parsing input
    - Performing Calculations
    - Servicing similar requests
    - Managing activities and sequences

Classes – Composite

- Usually used to model something that has properties and to support methods or behaviors expected of the modeled object.
- For example, we can model an automobile
  - What do we know about the automobile?
  - What can we do with the automobile?
  - What about while we are using the automobile?
    - Location, direction, speed, people on board

Polymorphism/Interface

- Define classes with same methods
- May substitute one class for another if they support the same methods.
  - Car, Motorcycle, Bicycle
  - Feet, Wheels, Tank treads

Polymorphism/Inheritance

- Create class objects that are special type of other objects.
- Vehicle
  - Motorize Road Vehicle
    - Car
      - Gocart
      - Convertible
      - Minivan
    - Truck
      - Light Duty
        - Pickup
        - Van
      - Heavy Duty
        - Flat bed
        - Van
        - Tractor/Trailer
    - Motorcycle

Program Flow

- Create class with a “main” method
- Should contain methods to initialize the program and its components
- Should contain methods to cleanup before stopping the program
- Should contain or call methods for the main processing loop.

Summary

- Came to learn Java and we spent most of the time on understanding what we are trying to accomplish and how to manage our efforts.
- In real life, the programming language used is a small, but critical part of any project.
- We discussed why Java is a good candidate.
- We discussed how the basic constructs of Java can be used to model our components and how we use them.