Final Assessment

Java Summative Assessment.pdf
80.9 KB

Java Fundamentals Marking Guide.xlsx
22.2 KB

Assessment Outline

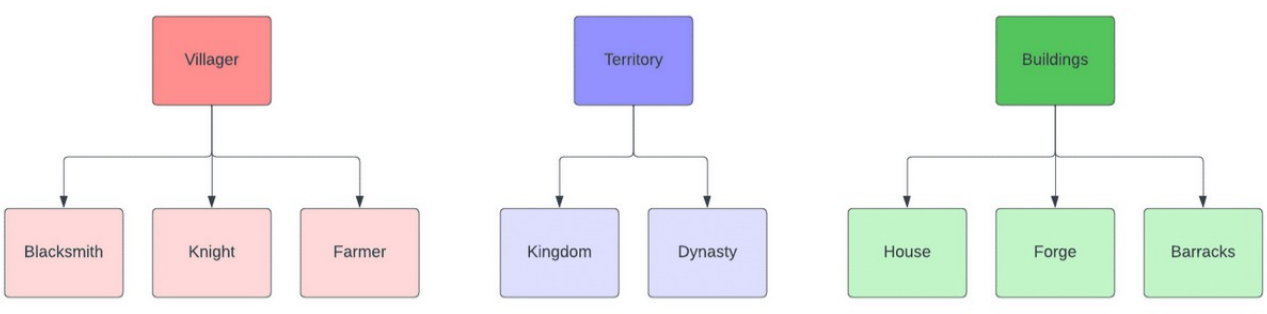
After looking at the Java Summative Assessment, you might be feeling a bit overwhelmed with where to start. This small guide will provide a jumping-off point to flesh out the basics without completing the assignment for you. Remember following your inspiration is a good thing here. The program only outlines some basic OOP requirements. Your final project could be a small Text Based Game, A town Simulator or anything you like.

Assessment Submission

To submit your assessment export your project from IntelliJ as a zipped folder. You can then submit your project on the submission form that follows this lesson.

First Steps

This small tutorial will cover building the Villager object. Defining what a Villager will look like in my project.



Object Chart

To break down my version of a Villager I want them all to have a:

- First Name (Given Name to identify individuals)
- Last Name (Family Name to track family tree)
- Age (Age of Villager)

Keeping things simple I am building this out as an Abstract Class as I don't want to be able to create a Villager without a profession.

Villager.java

```java
package Villagers;

public abstract class Villager {

    String FirstName;
    String LastName;
    int Age;

    public Villager(String firstName, String lastName, int age) {
        FirstName = firstName;
        LastName = lastName;
        Age = age;
    }

    public String getFirstName() {
        return FirstName;
    }

    public String getLastName() {
        return LastName;
    }

    public int getAge() {
        return Age;
    }

    public void setFirstName(String firstName) {
        FirstName = firstName;
    }

    public void setLastName(String lastName) {
        LastName = lastName;
    }

    public void setAge(int age) {
        Age = age;
    }
}
```

Villager Class

Knight

Expanding on the Villager, We are now creating a knight. The knight will add the Attributes of Strength (Randomly generated number) and a Weapon. First, I am creating an Enumerator of a list of weapons the knight could have. This is able to be easily updated with more weapons later.

Weapons.java

```java
package Villagers;

public enum Weapons {
    Sword,
    Mace,
    Club,
    Dagger,
    Lance
}
```

Weapon Enum

Knight Functions

Knight.java

```java
package Villagers;

import java.util.Random;
import java.util.Scanner;

public class Knight extends Villager{

    private Weapons weapon;
    private int strength;

    public Knight(String firstName, String lastName, int age) {
        super(firstName, lastName, age);
        generateStrength();
        selectWeapon();
        System.out.println("\nKnight created\n");
        print();
    }

    private void generateStrength() {...}

    private void selectWeapon(){...}

    public Weapons getWeapon() {
        return weapon;
    }

    public int getStrength() {
        return strength;
    }

    public void print(){
        System.out.println("Name: " + getFirstName() + " " + getLastName());
        System.out.println("Age: " + getAge());
        System.out.println("Strength: " + getStrength());
        System.out.println("Weapon: " + getWeapon());
    }

}
```

Knight Class

Building out the Knight class i am extending from the abstract villager class and adding my new attributes with their Getters as well as a print function to display the Stats of the Knight.

The generate strength function simply generates a number between 1 and 10. This value could be used to calculate the victor of a fight between knights.

```java
private void generateStrength() {
    Random rand = new Random();
    strength = rand.nextInt( origin: 1, bound: 10);
}
```

Generate strength function

```java
private void populateTerritory(int iterations){
    Scanner scanner = new Scanner(System.in);
    for(int i = 0; i <= iterations-1; i++){
        System.out.print("\nEnter first name: ");
        String fName = scanner.nextLine();
        System.out.print("Enter last name: ");
        String sName = scanner.nextLine();
        System.out.print("Enter age: ");
        int age = Integer.parseInt(scanner.nextLine());
        this.villagers.add(new Knight(fName,sName,age));
    }
}
```

Populate Territory

The Populate Territory function just calls for user input to fill out the three villager parameters, then calls the Knight constructor.

Enhancements

The provided tutorial can be enhanced in many ways. The villager class could have its own print statement overridden by the child classes. Knights could have a fight function that uses their strengths a determining factor in a fight.

Other villager types could provide enhancements to their respective kingdom.
Territories could go to war, Buildings could be lost.
A simple random number generator could control an enemy kingdom that makes a decision every time you do.

Follow your inspiration and if you ever reach a point where you aren't sure how to implement a feature the community Discord is always there ready to answer questions.