# FA21 CSE 8B Homework 5: Pokemon Go

Due Date: Wednesday, November 3, 11:59 PM

## Learning goals:

- Understand object-oriented programming (OOP) concepts through:
  - Objects (and object-oriented thinking)
  - Inheritance
  - Classes

**NOTE: This assignment should be completed INDIVIDUALLY. Pair programming is NOT allowed for this assignment.**

**IMPORTANT: You should NOT have to import any additional packages to complete this assignment. Any unnecessary imports may result in failure to compile on Gradescope.**

**THIS IS A LONG ASSIGNMENT, SO PLEASE START EARLY!!!**

---

## Coding Style (10 points)

**For this programming assignment, we will be enforcing the CSE 8B Coding Style Guidelines**. These guidelines can also be found on Canvas. Please ensure to have *COMPLETE* file headers, class headers, and method headers, to use descriptive variable names and proper indentation, and to avoid using magic numbers.

---

## Part 0: Getting started with the starter code (0 points)

1. Make sure there is no problem with your Java coding environment. If there are any problems, then review Assignment 1, or come to the office/lab hours before you start Assignment 4.
2. Download the starter code.
   a. If you are working on your local machine, then you can download the starter code from Piazza → Resources → Homework → `Assignment5.zip`. Download the starter code to a directory of your choice, then extract the `zip` file. It should have the following structure:

```
+-- starter/
|     +-- Berry.java              Edit this file
```

```
|     +-- Item.java              Edit this file
|     +-- PalPokemon.java        Edit this file
|     +-- Player.java            Edit this file
|     +-- Pokeball.java          Edit this file
|     +-- Pokemon.java           Edit this file
|     +-- WildPokemon.java       Edit this file
|     +-- WildWorld.java         Edit this file
|     +-- Assignment5.java       Edit this file
|     +-- NumberGenerator.java   DO NOT change
```

b. If you are working via UCSD Linux Cloud through your CSE 8B account, then use the commands below to copy the starter code to a new directory called `PA5`, to change your current directory to `PA5/starter`, and to print the files in the `starter` directory:

```
$ cp -r ~/../public/assignments/PA5 ~
$ cd ~/PA5/starter
$ ls
```

---

## Overview

In this assignment you will be implementing a game inspired by Pokemon Go. As you can see, there are a lot of files you need to edit. So make sure you follow the instructions below carefully.

In the game, there will be a class named `Pokemon`. There will be two subclasses of `Pokemon` class, which are `WildPokemon` and `PalPokemon`. Also, there will be a class called `Item`. Under the `Item` class, there will be two subclasses named `Berry` and `Pokeball`. These two items can help you to catch a wild pokemon to be your pal pokemon. You can use a `Pokeball` to catch, and use a `Berry` to enhance the catch rate. You can only use up to one berry before you throw a pokeball. There will be different kinds of pokeballs and berries. Details will be explained later.

You have another two classes called `Player` and `WildWorld`. A player can have a collection of pokeballs and only up to one berry. Inside each pokeball, there might be a pal pokemon. A

wild world has a collection of wild pokemons. The player's goal is to catch those wild pokemons with his collection of pokeballs.

You also have a class called `NumberGenerator`. This class is used to randomly generate an integer. The usage will be discussed later on.

Finally, you have a class called `Assignment5` that allows you to add testers and execute the program.

**IMPLEMENTATION TIP**: you should NOT change any data field or method signature in the starter code. As such, please observe the starter code and read the instructions below to make sure you understand what each field means before you start to implement.

---

## Part 1: `Item.java`

First, you need to implement the object class called `Item`. This is a super class and gets inherited by `Pokeball` and `Berry`.
The `Item` class should contain the following fields (all are provided in the starter code):

1. `public Item()` - Initialize name to be String "Item".
2. `public Item(String name)` - Initialize name with the given parameter
3. `public String getName()` - retrieves name
4. `public String setName(String name)` - mutates name with the given parameter

---

## Part 2: `Pokeball.java` and `Berry.java`

Those two classes are children of the `Item` class. The `Pokeball` class has additional `performance` and `palPokemon` member variables. The `Berry` class has additional `luckiness` and `speedBoost` member variables.

The Pokeball class should contain the following methods for you to implement:

1. `public Pokeball()` - initialize name with String "Pokeball"

2. `public Pokeball(String name, int performance)` - initialize the fields with given parameters
3. `public void getPerformance()` - retrieves performance
4. `public void setPerformance(int performance)` - mutates performance
5. `public void getPalPokemon()` - retrieves palPokemon
6. `public void setPalPokemon(PalPokemon palPokemon)` - mutates palPokemon
7. `public boolean isEmpty()` - return true if palPokemon is null. Otherwise return false

The Berry class should contain the following methods for you to implement:

1. `public Berry()` - Initialize name with String “Berry”
2. `public Berry(String name, int luckiness, int speedBoost)` - Initialize all member variables with the given parameters.
3. `public int getLuckiness()` - Retrieves luckiness
4. `public int setLuckiness(int luckiness)` - mutates luckiness
5. `public int getSpeedBoost()` - retrieves speedBoost
6. `public int setSpeedBoost(int speedBoost)` - mutates speedBoost

---

## Part 3: `Pokemon.java`

This is a super class and gets inherited by `PalPokemon` and `WildPokemon`. It has three fields. For the field `type`, it’s an attribute of a certain pokemon. For example, Pikachu is an electric type pokemon. Complete all constructors by initializing member variables with the given parameters. In the no-arg constructor, set String `name` to be “Pokemon”, String `sound` to be “Sound”, String `type` to be “Unknown”. Complete setters and getters.

---

## Part 4: `PalPokemon.java`

PalPokemon is a subclass of Pokemon. This represents a pokemon a player caught.

PalPokemon has a `pokeball` field. Since pal pokemon belongs to the player, it must be contained in a `pokeball`. Complete all constructors, setters, and getters. For the no-arg constructor, initialize `pokeball` to null.

---

## Part 5: `WildPokemon.java`

WildPokemon is a subclass of Pokemon. This represents a pokemon you might want to catch in the wild. It has three integer member variables: `power`, `speed` and `timesEscapedFromBall`.

The WildPokemon class should contain the following methods for you to implement:

1. `public WildPokemon()` - Initialize `power` to be 100, `speed` and `timesEscapedFromBall` to be 0.
2. `public wildPokemon(String name, String sound, String type, int power, int speed)` - Initialize all member variables with the given parameters.
3. Three setters and three getters.
4. `public boolean isFasterThan(int ballSpeed, Berry berry)` - return true if the speed of the wild pokemon is greater than the `ballSpeed` plus the `speedBoost` from berry. Otherwise, return false.
5. `public boolean canEscapeFromBall(Pokeball pokeball, Berry berry)` - return true if the `power` of the wild pokemon is greater than the performance of the pokeball plus the luckiness of the berry. Otherwise, return false.
6. `public boolean canBeCollectedBy(Berry berry, Pokeball pokeball, int ballSpeed)` - if the wild pokemon is faster than the `ballSpeed` plus the `speedBoost` from berry (use the method `isFasterThan`), return false. Then, check If the wild pokemon can escape from the ball. If it can escape, increment the `timesEscapedFromBall` and return false. In the end, return true.

---

## Part 6: `Player.java`

Player is a class that represents a player to play Pokemon Go. It has a member variable `pokeballs` that represents a list of pokeballs that can be used to catch wild pokemon or to

contain pal pokemon. It has a `berry` that will increase the likelihood of catching a wild pokemon.

The Player class should contain the following methods for you to implement:

1. `public Player(Pokeball[] pokeballs, Berry berry)` - initialize the member variables with the given parameters
2. Two getters and two setters
3. `public void showOff()` - It will print all pal pokemons collected inside each pokeball the player owns. It will print “Go my pokemons!!!” first. If any pokeball is empty, print “Pokeball x is empty” where x is the name of the pokeball. Otherwise, print “name: sound” where name is the name of pal pokemon and sound is the sound of pal pokemon contained in the pokeball. Here is an example:

```
Go my pokemons!!!
Pokeball UselessBall is empty
Charmander: char
Pokeball ultraball is empty
```

---

## Part 7: `WildWorld.java`

WildWorld is a class that represents the pokemon go wild world and there are many wild pokemons waiting to be caught. It has a member variable `wildPokemons` that represents the wild pokemons existing in this world.

The WildWorld class should contain the following method for you to implement:

1. `public WildWorld(WildPokemon[] wildPokemons)` - Initialize the member variable with the given parameter.
2. A setter method and a getter method.
3. `public void adventure(Player player)` - The player is trying to catch wild pokemons one by one with its pokeball. You may assume the length of `wildPokemons` is always the same as the length of the `pokeballs` the player has. The first pokeball is

used to catch the first wild pokemon, the second pokeball is used to catch the second wild pokemon, and so on.

If the current pokeball already contains a pal pokemon or the current wild pokemon is null, skip this pokeball and current wild pokemon. Otherwise, the player tries to catch the current wild pokemon. Generate an random integer with the method provided in NumberGenerator class. This random integer will be used as our ball speed.

Then check if the current wild pokemon can be collected by the player using the method `canBeCollectedBy` given by the generated ball speed, the berry of the player, and the current pokeball. If the wild pokemon can be collected, create a new pal pokemon with the same name, sound, and type, associate it with the current pokeball, and remove the wild pokemon from `wildPokemons` by setting the wild pokemon at the current position to null. Otherwise, if the wild pokemon can not be collected, simply move on to the next wild pokemon and next pokeball.

Note: Read the file NumberGenerator to understand how to use it. In addition, since we are using random variables, whether the player is able to catch the pokemon or not is non-deterministic. Therefore, you want to extensively test all methods used inside of this method to ensure correctness.

---

## Part 8: `Assignment5.java`

Inside Assignment5, a tester has already been written for you to test the functionality of canBeCollectedBy method in WildPokemon class. As before, you are encouraged to create as many test cases as you think to be necessary to cover all the edge cases. However, since there are so many methods in this assignment, we will not ask you to create test cases for each method. **To get full credit, please create *at least five* additional test cases that test all your different methods.** We suggest making some print messages in each of your test cases so that you will know which test case is failing. The `unitTests` method should `return true` only when all the test cases are passed, otherwise return false.

In addition, a method called simulation is provided to show how a player can play our Pokemon Go. This simulation will print the following to the console if your implementation is correct:

```
Go my pokemons!!!
Pokeball UselessBall is empty
Charmander: char
Pokeball ultraball is empty
```

Important: Make sure your setters and getters are correct! If your setters and getters are wrong, you will receive more points deducted than expected as the Autograder uses them extensively.

---

## Submission

**VERY IMPORTANT: Please follow the instructions below carefully and make the exact submission format.**

1. Go to Gradescope via Canvas and click on `PA5`.
2. Click the DRAG & DROP section and directly select the required files (**`Assignment5.java`**, **`Berry.java`**, **`Item.java`**, **`PalPokemon.java`**, **`Player.java`**, **`Pokeball.java`**, **`Pokemon.java`**, **`WildPokemon.java`**, **`WildWorld.java`**). Drag & drop is fine. **Please make sure you don't submit a zip. Make sure the filenames are correct.**
3. **You can resubmit an unlimited number of times before the due date.** Your score will depend on your final submission, even if your former submissions have a higher score.
4. The autograder is for grading your uploaded files automatically. Make sure your code can compile on Gradescope. You will receive 0 out of 80 on Autograder if your code does not compile on Gradescope.

   **NOTE: The Gradescope Autograder you see is a minimal autograder.** For this particular assignment, it will only show the compilation results and the results of a few testers. After the assignment deadline, a thorough Autograder will be used to determine the final grade of the assignment. **Thus, to ensure that you would receive full points from the thorough Autograder, it is your job to extensively test your code for correctness via `unitTests` and `main`.**