Homework 11 - PokeBattle!

Problem Description:

You have landed your dream job at an small video game company in Kyoto, and your first task is to create a GUI for battle sequences in the company's newest game- Pokemon.

Solution Description:

You will be provided with a Pokemon.java class which serves as a template for Pokemon, as well as a Move.java class.

You will create a file, `PokeBattle.java` which will extend `Application`, and serve as the GUI. This will also contain any button listeners, helper functions, and logic to support your application.

Static Requirements

- Your application must contain two major panes, each representing a Pokemon (i.e. the user Pokemon and the opponent Pokemon)
  - Each Pokemon pane should contain an image (see `ImageView` and `Image`) to represent the Pokemon (the image is entirely up to you, as long as it visually represents a unique Pokemon).
  - Each Pokemon pane must contain an information pane which will represent:
    - Pokemon's name
    - Pokemon's level
    - Pokemon's type
    - Pokemon's current health which must be dynamically updated as `Moves` are used (Hint: consider JavaFX's `ProgressBar` class)
  - The user Pokemon pane must also contain four buttons for user input (see dynamic requirements below), these are:
    - **FIGHT**
    - **BAG**
    - **POKEMON**
    - **RUN**

Dynamic Requirements

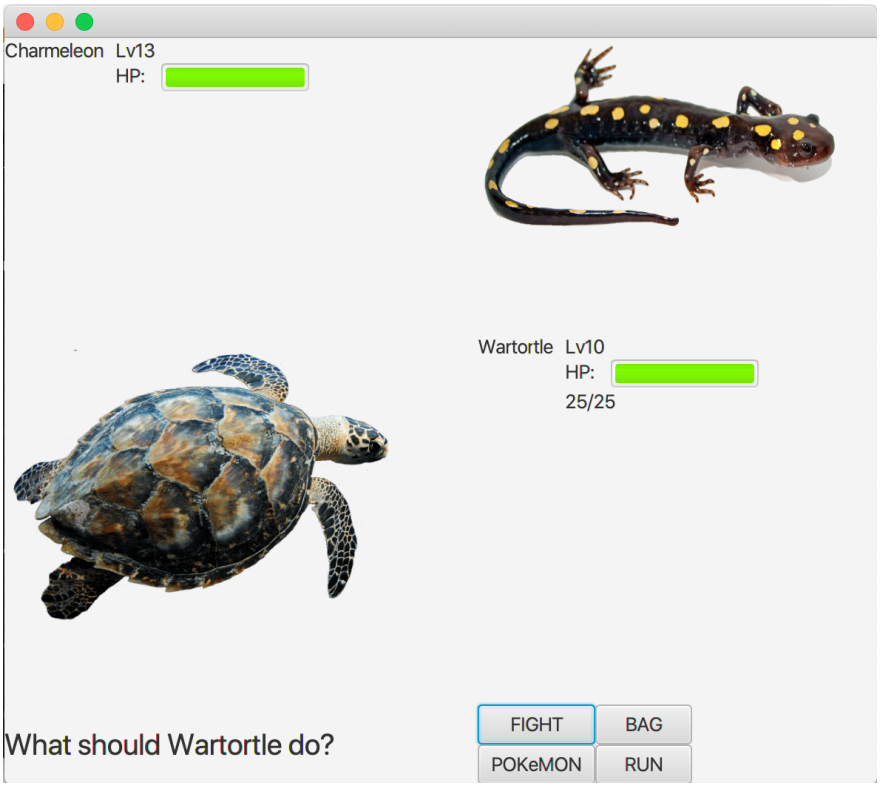
- The primary dynamic functionality in your application involves attacking each other via `Moves`, selected through the **FIGHT** button
  - When clicked, **FIGHT** must present the user with four `Moves` to choose from
    - **FIGHT** as well as a means to select a `Move` can be implemented with any UI element (Hint: `Button` class)
  - The user will then select a `Move` via a UI element, which will inflict some amount of damage on the opponent Pokemon
    - The exact amount of damage inflicted should be calculated using **at least two** instance variables from each Pokemon
    - You may use the Pokemon's `type`, `level`, `currentHP`, `atk`, `maxHP`, etc for calculating damage, provided there is some unique calculation involving **at least two** instance variables
    - Sophisticated algorithms for calculating damage may receive bonus points (see bonus options below)
  - Upon selecting an attack, the following must occur:
    - The opponent's HP is reduced and is reflected in the UI element representing their HP (i.e. visual representation or text)
    - The opponent will similarly attack the user Pokemon. The opponent Pokemon's `Move` may be determined randomly or algorithmically
    - The user's Pokemon's HP should be reduced and be reflected in the UI element representing its HP

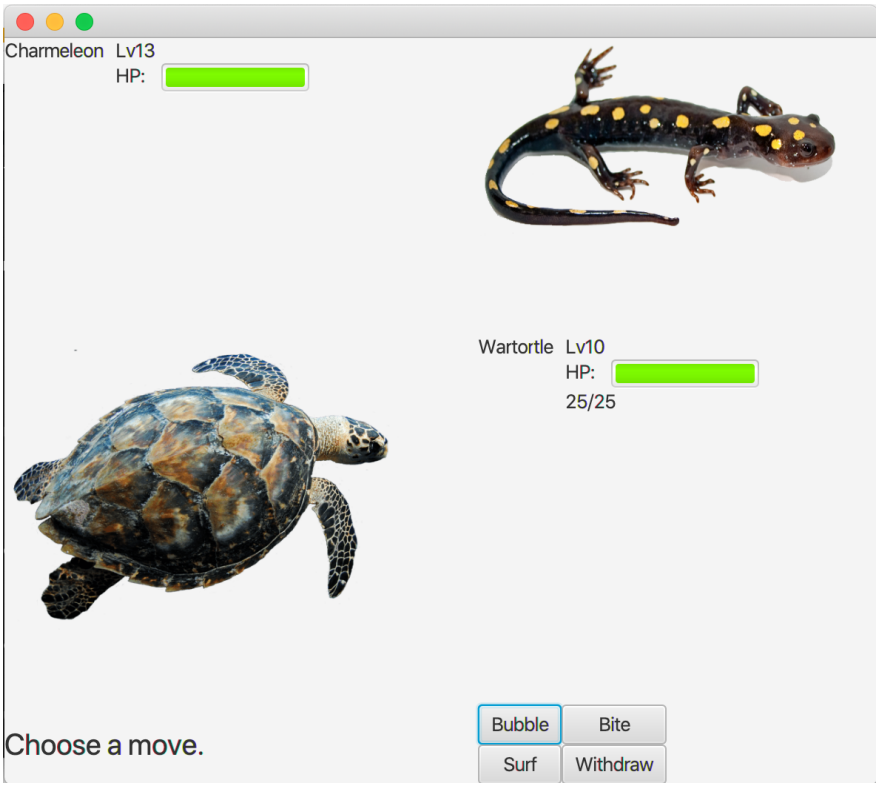
        * The application must return to the same state, with **FIGHT**, **BAG**, **POKEMON** and **RUN** buttons
        * *NOTE* to fulfill the base requirements for this assignment, these can all be done instantaneously without a specified delay.
    - If the user's Pokemon or the opponent Pokemon has its HP reduced to less than or equal to 0, the application should exit
        * If the user's HP is reduced to zero, exit the program (**this is the one homework where you can use System.exit()**)
        * If the opponent HP is reduced to zero, exit the program (hint: `System.exit()` again)
- Additionally, pressing **RUN** must must exit the program (as though being closed).
- **POKEMON** and **BAG** buttons do not need to do anything (aside from being present) to get full credit (see bonus options)

Bonus Options

- As this homework is relatively open-ended, up to 50 bonus points will be made available for, but not limited to:
    - Sophisticated algorithms for determining attack damage
    - Sophisticated visual displays to notify that an attack occurred (such as the Pokemon flashing/moving or other `Image` and `ImageViews`s appearing temporarily)
    - Text notifications such as "[user Pokemon name] fainted!" and "[user Pokemon name] used [move]! and"Got away safely!" with appropriate delays in order to understand the text.
    - Aesthetically interesting UI elements (such as background colors, interesting borders, background images, etc)
    - Features involving **POKEMON** and **BAG** buttons
        * Create an array of Pokemon, and allow switching between Pokemon (**POKEMON** button)
        * Create items, possibly an `Item.java` class, to modify instance variables of either user or opponent Pokemon (increasing HP, changing level, changing name)
        * Any other creative ideas involving these buttons are eligible for extra credit, subject to TA discretion
    - Any additional features beyond **static requirements** and **dynamic requirements** may receive extra credit, subject to TA discretion
- These bonus points will count towards overall homework grade total

Example of Screens





Rubric

- [10] Starting screen has 4 buttons: {FIGHT, BAG, POKEMON, RUN}
- [5] Starting screen has image representing user's Pokemon
- [5] Starting screen has image representing opponenet's Pokemon
- [10] Starting screen has the specified information regarding the user's Pokemon
- [10] Starting screen has the specified information regarding the opponent's Pokemon
- [10] Clicking **FIGHT** presents the user with 4 moves to choose from
- [10] After selecting a move, the opponent's HP is reduced
- [5] The damage is calculated using at least 2 instance variables of the Pokemon
- [10] After selecting a move, the opponent attacks the user's Pokemon and its HP is reduced
- [10] After selecting a move, the original 4 buttons are present: {FIGHT, BAG, POKEMON, RUN}
- [5] If the user's HP is 0 or less, the program exits
- [5] If the opponenet's HP is 0 or less, the program exits
- [5] If the RUN button is pressed, the program exits
- [up to 50] EXTRA CREDIT

The Checkstyle cap for this assignment is **50 points**. This means that up to fifty points can be lost from Checkstyle errors. JavaDocs will be required, including for any helper methods or extra classes. To check for JavaDocs with checkstyle, make sure to include the - A tag.

```
$ java -jar checkstyle-6.2.2.jar - A *.java
```

We reserve the right to adjust the rubric, but this is typically only done for correcting mistakes.

Running and Testing

Because Java 11 does not come with JavaFX built in, you will need to download an SDK and use a different compiling command.

- Follow this link to download the necessary SDK: https://gluonhq.com/products/javafx/
- You should be able to download a zipped folder: `javafx- Sdk-11.0.2`. Extract the contents to the directory holding your HW10 code.
- Compile your application with the following command:

```
javac --module-path javafx-sdk-11.0.2/lib --add-modules=javafx.controls *.java
```

Note: wildcard compile works if all files needed are in the same directory, otherwise, replace * with `filename.java` - Launch your application with the following:

```
java --module-path javafx-sdk-11.0.2/lib --add-modules=javafx.controls PokeBattle
```

Alternatively you may use an IDE such as Intellj for compiling and running, as it will import and add to classpath any supporting SDK you may need. Feel free to come to office hours or ask on Piazza for assistance.

Allowed Imports

You are allowed to import and use **any** `Class` provided by Java (especially JavaFX-related classes) Please come to office hours or ask on Piazza for suggestions on useful classes.

Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach. For that reason, do not use any of the following in your final submission:

- var (the reserved keyword)
- System.exit() is allowed for this homework only

Additionally, you may **NOT** use any markup (.xml or .fxml) or drag- And-drop editors to generate .xml/.fxml code for creating the UI.

Collaboration

Collaboration Statement

To ensure that you acknowledge a collaboration and give credit where credit is due, **we require that you place a collaboration statement as a comment at the top of at least one Java file that you submit**. That collaboration statement should say either:

I worked on the homework assignment alone, using only course materials.

or

In order to help learn course concepts, I worked on the homework with [give the names of the people you worked with], discussed homework topics and issues with [provide names of people], and/or consulted related material that can be found at [cite any other materials not provided as course materials for CS 1331 that assisted your learning].

Turn- In Procedure

Submission

To submit, upload the files listed below to the corresponding assignment on Gradescope:

- Pokemon.java
- PokemonBattle.java
- Move.java
- any additional files you have created (ensure that these compile, as TAs will compile and test locally)
  - this includes **all image files**
- Do **not** include any SDK files- TAs will have these locally for compiling your code.

Make sure you see the message stating "HW10 submitted successfully". There is no autograder for this homework- it will be manually graded.

You can submit as many times as you want before the deadline, so feel free to resubmit as you make substantial progress on the homework. We will only grade your last submission: be sure to **submit every file each time you resubmit**.

Gradescope Autograder

There is no autograder for this assignment. Alternatively, please come to office hours to allow TAs to review your progress if desired.

Important Notes (Don't Skip)

- Non- Compiling files will receive a 0 for all associated rubric items
- Test your code in addition to the basic checks on Gradescope
- Submit every file each time you resubmit
- Read the "Allowed Imports" and "Restricted Features" to avoid losing points
- Check on Piazza for a note containing all official clarifications