Assignment for Week 10: User Interactions

Goals

In the first part you will practice user interaction via the console.
In the second part you will modify the program by implementing GUI and graphics view of the program state and adding interactive user controls in the form of buttons and mouse manipulations.

The Two Games

You will implement one of the following two games:

The War Game

As the bomb ifs falling from the sky, the defenders try to shoot it in mid-air, so it would not cause any damage. The game starts with a plane loaded with some given number of bombs, but only one bomb is dropped at a time, the second bomb dropped only after the first one has hit the earth or has been destroyed. The game ends after all bombs have been dropped and either have hit the earth, or have been destroyed. At the end of the game the total bombs dropped and the number of bombs destroyed in mid-air is reported.

The Peace Game

At a peace celebration, balloons are released into the air and they start floating upwards. The members of the Green Force start shooting darts at the balloons, making sure the balloons burst and fall to the ground in the vicinity of the celebration, where they will be collected by the Green Force. This way, they will not float away and eventually cause harm to birds who may eat them and choke. The game starts with the celebrants holding a given number of balloons, and release the balloons one at a time, with the next balloon being releases after the first one has been either destroyed, or has floated outside of the range of the Green Force. The game ends after all balloons have been released and have either been destroyed or have floated away. At the end of the game the total balloons released and the number of balloons destroyed is reported.

Part 1: Console Interactions

1. The given `class Circle` represents either one bomb or one balloon. You may select any color you choose for this object (as long as it can be clearly visible in the graphics window when painted). The class comes only with the default constructor (with no fields set) and a paint method. Follow the design recipe to develop the following methods:

   - `void erase(BufferedPanel window)` and test is by painting and erasing a bomb (balloon) in the graphics window
   - `void fall(int dy)` or (if you are programming is the Peace Game) `void rise(int dy)` which changes the location of this object by the specified distance.
   - `boolean hit(Point p)` which determines whether the shot aimed at the given point hit the bomb (balloon).

2. Design and implement as a Java class, following the design recipe — the `class Game` which records the progress of the game. The class needs to keep track of the following information:

   - the original number of bombs (balloons)
   - the number of bombs (balloons) released
   - the number of bombs (balloons) hit
   - the speed of the falling (rising) — given as an integer
   - the currently falling bomb (rising) balloon
   - the height of the graphic window

   The following methods will be needed to play the game:

   - Initialize the game by asking the user to specify the number of bombs (balloons), and the speed of falling (rising).
   - Method which produces a random shot: it generates a Point.
   - Method `oneBomb` or `oneBalloon` which simulates the dropping (rising) of one object. At each time *t*ick it checks whether the random shot was a hit. If it was a hit, the method destroys the bomb (balloon) and reports the hit. Otherwise, the method moves the bomb (balloon) by the distance determined by the speed value. If the resulting location is outside of the range, it reports a miss, otherwise it repeats the simulation.

- Method `playGame` which simulates the overall game: It initializes the game, then proceeds to release one object at a time (`oneBomb` or `oneBalloon`) and record whether this was a hit or miss. Once all objects have been released and accounted for, report the game results.

Technical details

The `class Game` needs to interact with the `JPTConsole`. To make this possible you need to do two things:

- `import edu.neu.ccs.console.*;`
  statement must appear before the class decalaration
- `class Game implements ConsoleAware`
  though no further actions is required to complete the implementation.
- The following methods (available for all primitive types) can be used to process user input:
  - `int x = console.in.demandInt("Next number:");`
    includes a prompt for the user, demand must be fulfilled
  - `double y = console.in.demandDouble("Next number:", 100.0);`
    includes a prompt and a default value that will be used, if the user hits return without typing in anything

  -
    ```
    try
    {int x = console.in.requestInt("Next number:");}
    catch (CancelledException e)
    {console.out.println("No input given");}
    ```
    hitting the return without typing any input raises the exception