P05 Toy Saga II

Overview

Way back in P03 we had you refactor code that you previously wrote to follow object-oriented design. Again (for the last time) we will revisit our trusty Toy Saga code and alter it to take advantage of inheritance, interfaces, and casting. After completing this assignment, it should retain the original functionalities plus a couple of new features! For instance, in this new version toys can move on their own and we can change the scene from daytime to nighttime.

Learning Objectives

The goals of this assignment include:

- Implementing an interface
- Organizing your code in an OOP-oriented fashion that utilizes inheritance relationships between classes
- Practicing overriding methods from a parent class or interface
- Experiencing polymorphism!
- Deepening your knowledge of the processing graphic library

Grading Rubric

5 points	**Pre-Assignment Quiz:** The P05 pre-assignment quiz is accessible through Canvas before having access to this specification by **11:59PM CT on Sunday March 10**.
+2.5 points	**5% BONUS Points:** Students whose final submission to Gradescope has a timestamp earlier than **4:59 PM CT on Wednesday, March 13** and passes ALL the immediate tests will receive an additional 2.5 points toward this assignment's grade on gradescope, up to a maximum total of **50** points.
15 points	**Immediate Automated Tests:** Upon every submission of your assignment to Gradescope, you will receive immediate feedback from automated grading tests about whether specific parts of your submission conform to this write-up specification. If these tests detect problems in your code, they will attempt to give you some feedback about the kind of defect that they noticed. Passing all immediate automated tests does NOT guarantee full credit for the assignment.
20 points	**Additional Automated Tests:** When your manual grading feedback appears on Gradescope, you will also see the feedback from these additional automated grading tests. These tests are similar to the Immediate Automated Tests, but may test different parts of your submission in different ways.
10 points	**Manual Grading:** Human graders will review the commenting, style, and organization of your final submission. They will be checking whether it conforms to the requirements of the CS300 Course Style Guide. You will NOT be able to resubmit corrections for extra points, and should therefore carefully review the readability of your code with respect to the course style guide.
50 points	**MAXIMUM Total Score**

Assignment Requirements and Notes

(Please read carefully!)

Pair Programming and Use of External Libraries and Sources

- Pair programming is **ALLOWED** for this assignment.
- Any source code provided in this specification may be included verbatim in your program without attribution.
- All other sources must be cited explicitly in your program comments, in accordance with the Appropriate Academic Conduct guidelines.
- Any use of **ChatGPT** or other large language models (LLM) must be cited AND your submission MUST include a file called `log.txt` containing the full transcript of your

usage of the tool. Failure to cite or include your logs is considered academic misconduct and will be handled accordingly.

- You are only allowed to **import** or **use** the libraries listed below in their respective files **only**.
- **The ONLY external libraries** you may use in your submitted file are:

ToySaga.java	java.io.File, java.util.ArrayList, processing.core.PApplet, processing.core.PImage
TeddyBear.java	processing.core.PApplet
GraphicObject.java	processing.core.PImage
Hoverball.java	processing.core.PApplet, java.io.File

P05 Assignment Requirements

- **NONE** of the callback methods (`setup()`, `draw()`, `mousePressed()`, `mouseReleased()`, and `keyPressed()`) defined later in this write-up should be called explicitly in this program.
- You MUST NOT add any additional fields either instance or static to your program, and any public methods either static or instance to your program, other than those defined in this write-up.
- DO NOT add any statement to the provided `main()` method, other than the call to `PApplet.main("ToySaga")`.
- The automated grading tests in gradescope are **NOT using the full processing library** when grading your code. They only know about the following fields and methods (referenced directly from this assignment). The only methods that you are allowed to use from the `PImage` and `PApplet` classes defined in the processing core library are the following:
  - **two** fields from `PImage`: int `width`, and int `height`.
  - **three** methods from `PApplet`: PImage `loadImage(String)`,
    void `image(PImage,int,int)`,
    void `main(String)` OR those that we **explicitly** mention in this write-up.

  If you are using any other fields, methods, or classes from the `PImage` or `PApplet` classes NOT **explicitly** mentioned in this write-up, this will cause problems for the automated grading tests.

CS300 Assignment Requirements

This section is VALID for ALL the CS300 assignments

- If you need assistance, please check the list of our Resources.
- You MUST NOT add any additional fields either instance or static, and any public methods either static or instance to your program, other than those defined in this write-up.
- You CAN define local variables (declared inside a method's body) that you may need to implement the methods defined in this program.
- You CAN define **private** methods to help implement the different public methods defined in this program, if needed.
- Your assignment submission must conform to the CS300 Course Style Guide. Please review ALL the commenting, naming, and style requirements.
  - Every submitted file MUST contain a complete file header, with accordance to the CS300 Course Style Guide.
  - All your classes MUST have a javadoc-style class header.
  - All implemented methods including the main method MUST have their own javadoc-style method headers, with accordance to the CS300 Course Style Guide.
  - Indentation should be 2 spaces and NOT 4 spaces.
  - The maximum width line should be 100.
- If starter code files to download are provided, be sure to remove the comments including the `TODO` tags from your last submission to gradescope.
- Avoid submitting code which does not compile. Make sure that ALL of your submitted files ALWAYS compile. A submission which contains compile errors won't pass any of the automated tests on gradescope.
- Run your program locally before you submit to Gradescope. If it doesn't work on your computer, it will not work on Gradescope.
- You are responsible for maintaining secure back-ups of your progress as you work. The OneDrive and GoogleDrive accounts associated with your UW NetID are often convenient and secure places to store such backups. Aspiring students may try their hands at version control.
- Be sure to submit your code (work in progress) of this assignment on Gradescope both early and often. This will 1) give you time before the deadline to fix any defects that are detected by the tests, 2) provide you with an additional backup of your work, and 3)

help track your progress through the implementation of the assignment. These tests are designed to detect and provide feedback about only very specific kinds of defects. **It is your responsibility to implement additional testing to verify that the rest of your code is functioning in accordance with this write-up.**

- You can submit your **work in progress (incomplete work) multiple times** on gradescope. Your submission may include methods not implemented or with partial implementation or with a default return statement.

1 Getting Started

1.1 Create the Project

1. Create a new project in Eclipse, and name it something like "P05 Toy Saga II".

   a. Ensure this project uses Java 17. Select "JavaSE-17" under "Use an execution environment JRE" in the New Java Project dialog box.

   b. Do **NOT** create a project-specific package; use the default package.

2. Download the following Java source files from the p05 assignment page on Canvas:

   - `Callout.java`
   - `MouseListener.java`
   - `ToySaga.java`
   - `SwitchButton.java`

   Note that the SwitchButton.java file shows compiler errors when you add it to your project, at this level. These errors will be resolved after you complete the last instructions in the 1.6 subsection.

3. Create three (7) Java source files within your project's src folder:

   - `Car.java`
   - `Drawable.java`
   - `Movable.java`
   - `Toy.java`
   - `TeddyBear.java`
   - `Hoverball.java`
   - `GraphicObject.java`

   Only the `ToySaga` class should contain a `main()` method.

1.2 Download JAR File

We prepared a jar file named `core.jar` that contains an interface with the Processing library to help you build this assignment.

1. **Download** the `core.jar` file available on the P05 assignment page on Canvas and copy it into the project folder that you just created.
2. Then, **add** this jar file to your project's Java build path with the following steps, provided for Eclipse users. Instructions on how to add a jar file to the build path of a Java project on IntelliJ can be found here.

    a. Right-click on this file in the "Package Explorer" within Eclipse, choose "Build Path" and then "Add to Build Path" from the menu. If the .jar file is not immediately visible within Eclipse's Package Explorer, try right-clicking on your project folder and selecting "Refresh".

    b. **For Chrome users on MAC**, Chrome may block the the jar file and incorrectly reports it as a malicious file. To be able to copy the downloaded jar file, Go to "chrome://downloads/" and click on "Show in folder" to open the folder where your jar file is located.

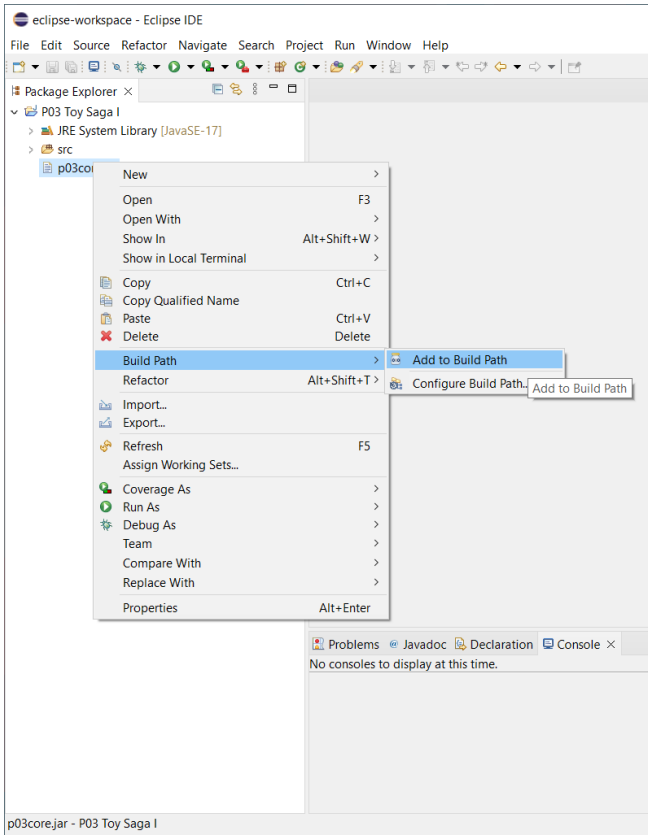
    c. **If the "Build Path" entry is missing** when you right click on the jar file in the "Package Explorer", follow the next set of instructions to add the jar file to the build path:

        1. Right-click on the project and choose "Properties".
        2. Click on the "Java Build Path" option in the left side menu.
        3. From the Java Build Path window, click on the "Libraries" Tab.
        4. You can add the `p05core.jar` file located in your project folder by clicking "Add JARs..." from the right side menu.
        5. Click on "Apply" button.

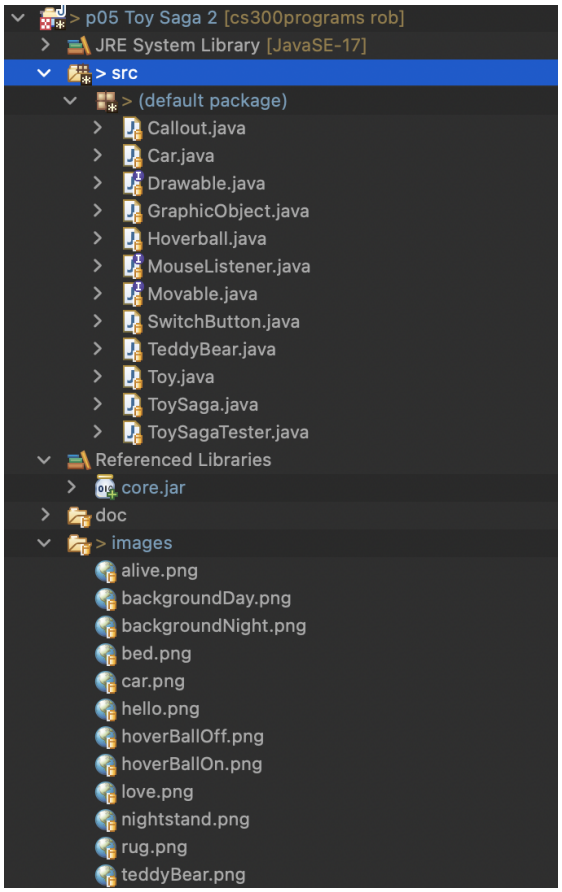
        This operation is illustrated in Fig. 1 (a)).

1.3 Download Image Files

1. Download the `images.zip` file from the P05 assignment page on Canvas and unzip it. It contains twelve images: the background images for day and night (`backgroundDay.png` and `backgroundNight.png`), the images of seven furniture and toy objects (`bed.png`, `hoverBallOn.png`, `hoverBallOff.png`, `car.png`, `nightstand.png`, `rug.png`, and `teddyBear.png`), and three callout messages (`alive.png`, `hello.png`, `love.png`).
2. Add the unzipped folder to your project folder in Eclipse, either by importing it or drag-and-dropping it into the Package Explorer directly. Make sure you **unzip** the file before continuing; if you try to use the zip file directly, it won't work.



(a) Adding `p03core.jar` to build path (same process for p05, but with core.jar)



(b) P05 package explorer

Figure 1: Getting Started - P05 Package Explorer

3. The organization of your P05 project through Eclipse's package explorer after completing this step is illustrated by Fig. 1 (b).

1.4 Test Project Setup

Edit the `ToySaga` class to inherit from `PApplet`. *You must do this in order for the window the appear.* Add the following line to the main method in this class:

```
PApplet.main("ToySaga"); // starts the application
```

If you were to run the program successfully at this point, you should have a tiny window with the title, "P5 Toy Saga v2.0" and a grey background and no errors.

Now let's add some code! The following code. First, add the two data fields `drawingObjects` and `mode`. Next, add some settings to change the properties of the GUI.

1.5 Writing simple interfaces

Now we will write the `Movable` and `Drawable` interfaces. `Movable` has one method that takes no parameters and returns nothing, `move()`. `Drawable` has one method that takes no parameters and returns nothing, draw(). See Javadocs for more details.

1.6 ToySaga remaining fields and methods

- Add the remaining data fields as defined in the javadocs for the `ToySaga` class.
- Add the callback method `settings()` to be overridden as follows.

```
/**
* Sets the size of the display window of this graphic application
*/
 @Override
 public void settings() {
   this.size(800, 600);
 }
```

- Add the callback methods `setup()`, `draw()`, `mousePressed()`, `mouseReleased()`, and `keyPressed()`. These methods are predefined in the `PApplet` superclass and we'll be overridden in the `ToySaga` subclass.
- Add the following GUI settings at the beginning of the `setup()` method.

```
// sets the title and graphic environment properties of the display window
this.getSurface().setTitle("P5 Toy Saga v2.0");
this.textAlign(CENTER, CENTER);// horizontal alignment: center, vertical alignment: center
this.imageMode(CENTER);// interprets the second and third parameters of image() as the
                       // image's center point.
this.rectMode(CORNERS); // interprets the first two parameters of rect() as the location
                        //  of one corner, and the third and fourth parameters as the
                        // location of the opposite corner.
this.focused = true;// sets the processing program to be focused (true), meaning that
                    // it is active and will accept input from mouse or keyboard
```

- Next, in the `setup()` method, set the starting mode to DAY_MODE, load in the background image related to the DAY_BACKGROUND image filename, and instantiate the ArrayList of `DrawableObjects` to an empty list.

- Add the `draw()` method, which draws the background image and calls each drawableObject's draw() method.
- If you run your code at this point, you should get a window that shows the familiar Toy Saga scene.
- Add the `mousePressed()` method, which traverses the `DrawableObjects` arraylist and calls the `onClick()` method on every instance of `MouseListener` stored in the list.
- Add the `mouseReleased()` method, which traverses the `DrawableObjects` arraylist and calls the `onRelease()` method on every instance of `MouseListener` stored in the list.
- Your code should compile without errors. You can notice the importance of the use of interfaces. We have been able to implement the above callback methods even though none of the instantiable classes is yet implemented. No further changes will be made to the `draw()`, `mousePressed()`, and `mouseReleased()` in the rest of the assignment.
- Add the `keyPressed()` method and implement pressing the d-key and the n-key behaviors, as described in the javadocs of the keyPressed() method. You can then run the code, press the d-key and the n-key and watch the background image of the ToySaga application switching from day to night scene.

2 Instantiable Classes

2.1 GraphicObject

We will now write the `GraphicObject` class, which represents any object drawn to the screen. As such, it has fields for the current ToySaga object, PImage, and x and y positions. Implement the `GraphicObject` class in accordance to the Javadocs.

Then, from within the `ToySaga.setup()` method, set the processing of the GraphicObject class to this ToySaga object, and add four GraphicObjects to the list: a bed, a nightstand, and a rug, according to the todos provided below. By the way, set the processing for the SwitchButton class and add a switch button to the drawableObjects list.

```
// within the ToySaga.setup() method
// TODO set the Processing for the SwitchButton class to be this ToySaga object

// TODO set the Processing for the GraphicObject class to be this ToySaga object

// TODO create a new SwitchButton at position (565, 20) and add it
// to the drawableObjects list

// TODO create a new GraphicObject BED at position (520, 270) and add it
// to the drawableObjects list
```

```
// TODO create a new GraphicObject RUG at position (220, 370) and add it
// to the drawableObjects list

// TODO create a new GraphicObject NIGHTSTAND at position (325, 240) and add it
// to the drawableObjects list
```

If you run now your application, you can see the furniture GraphicObjects and the SwitchButton object added to the toy saga scene. You can click the switch button and watch the background of the scene switching from day to night mode and vice versa.

2.2 Toy Class

The `Toy` class inherits from `GraphicObject` and implements the `MouseListener` (which was provided) and Movable (which you just wrote) interfaces. It has ONE private instance field: boolean `isDragging`. Implement the constructors and methods described in the Javadocs for `Toy`.

2.3 Car

We are finally ready to implement some toys. Each toy (Car, TeddyBear, and Hoverball) inherit from the `Toy` class.
As you implement any of these toy classes, you can go and update the `ToySaga.keyPressed()` method to enable pressing a specific key to create a new specific toy (car, teddybear, or hoverball) and add it to the scene (arraylist drawableObjects) if the maximum number of toys was not reached.

The toys will move on their own once the scene turns from day to night. At daytime, the toys must stop moving. The toys are always able to be dragged during the day mode.
The fields for each class differ, as does their movement behavior. The `Car` class defines the three following fields:

- `absoluteSpeed`: private static (NOT final) field of type int initialized to 8. It represents the absolute moving speed of cars.
- `speed`: private instance (NOT final) field of type int. It represents the current horizontal movement speed of the toy.
- `isMovingRightward`: private instance (NOT final) field of type boolean. It indicates the movement direction of the car object (true mean the car is travelling to the right, false means it is travelling leftward). When the toy saga mode is night mode, When the car moves, its position is updated by its speed. Once it hits a wall, it flips its move direction. It begins to travel in the other direction.

- The details of fields and methods for the `Car` class are in the Javadocs.
- The following is the implementation of the helper method `drawCarNightMode` which draws the car when moving at night.

```
/**
 * Provided method to draw a car at night with respect to its move direction
 */
private void drawCarNightMode() {
  toySaga.pushMatrix();
  toySaga.rotate(0.0f);
  toySaga.translate(x, y);
  if (!isMovingRightward) {
    toySaga.scale(-1.0f, 1.0f);
  }
  toySaga.image(image, 0.0f, 0.0f);
  toySaga.popMatrix();
}
```

2.4 Hoverball

`Hoverball` has no fields. When it moves at night mode, its y-position changes according to the code below:

```
    int dY = Math.round(6 * PApplet.sin(toySaga.frameCount * 0.1f));
```

Implement the constructor and methods of the `Hoverball` class with accordance to the details provided in the Javadocs.

2.5 TeddyBear Class

- `TeddyBear` has a float indicating its rotation amount, boolean indicating its rotationDirection, Callout that keeps track of what it says.
- When the TeddyBear moves at night, it wiggles from left to right and a Callout appears with a random message. And its move method should utilize the PApplet.radians() method.
- Implement the constructor and methods of the `TeddyBear` class with accordance to the details provided in the Javadocs.
- The helper method to draw a TeddyBear at night is provided in the following.

```java
/**
 * Provided method to draw this talking TeddyBear at night with respect to its moves
 */
private void drawTeddyBearNightMode() {
  move();
  toySaga.pushMatrix(); // Save the current transformation matrix
  toySaga.translate(x, y); // Translate to the teddy bear's position
  toySaga.rotate(rotation * PApplet.PI / 2); // Apply rotation
  if (toySaga.getMode() == "NIGHT") {
    toySaga.image(callout.image, 20f, -90f);
  }
  toySaga.image(image, 0.0f, 0.0f); // Draw the image at the rotated position
  toySaga.popMatrix(); // Restore the previous transformation matrix
}
```

2.6 Remaining ToySaga Methods

1. You will now complete any remaining methods in the Javadocs for each class.

2. After completing these methods, you should be able to place and drag toys, switch the scene from day to night, and watch as the toys move on their own! A picture of the application after adding some toys during nighttime is shown in Fig. 2, and a demo video is linked on the Canvas assignment page under **Demo Video**.

Figure 2: ToySaga2 - Finished Product

3 Assignment Submission

Congratulations on finishing this CS300 assignment! After verifying that your work is correct, and written clearly in a style that is consistent with the CS300 Course Style Guide, you should submit your final work through Gradescope. The only eight (8) files that you must submit are:

- `Car.java`
- `Drawable.java`
- `Hoverball.java`
- `Movable.java`
- `GraphicObject.java`
- `Toy.java`
- `TeddyBear.java`
- `ToySaga.java`

Your score for this assignment will be based on your **"active"** submission made prior to the assignment due date of Due: **9:59PM CT Thu March 14**. The second portion of your grade for this assignment will be determined by running that same submission against additional offline automated grading tests after the submission deadline.