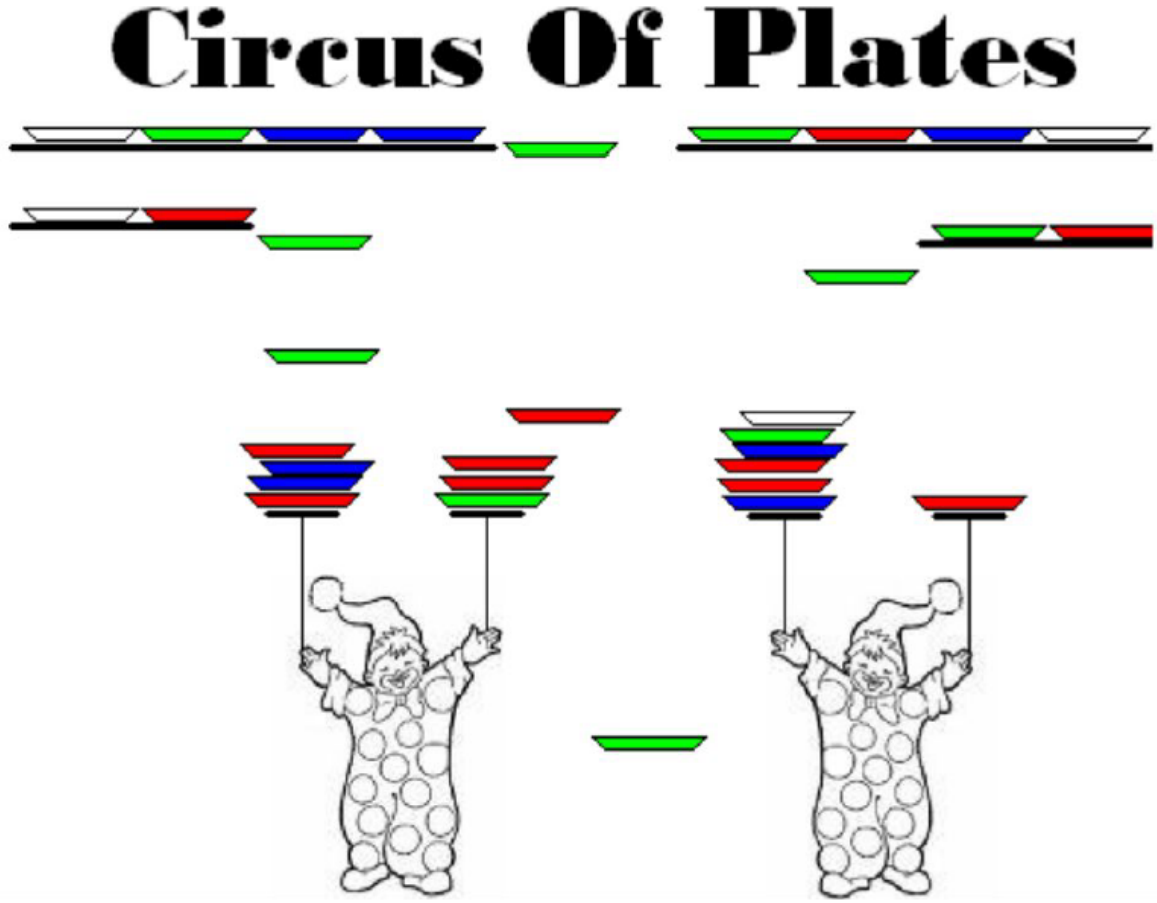
Circus of Plates – Game



Description

It is single player-game in which each clown carries two stacks of plates, and there are a set of colored plates queues that fall and he tries to catch them, if he manages to collect three consecutive plates of the same color, then they are vanished and his score increases. You are free to put rules ending the game.

Tasks

- You should not support only plates; you should support other shapes (you should have a class Shape). The shapes classes should be dynamically loaded at the start of the execution from a specific folder. You should support at least two shapes.
- The user gets a point when he collects three consecutive shapes from the same color (even if they are different shapes).
- You should use (at least) the following ten patterns in your design:

1. Singleton
2. Factory or Pool
3. Iterator
4. Dynamic Linkage
5. Snapshot
6. State
7. Strategy
8. Flyweight
9. Observer
10. and choose another one you feel it suits your design.

This assignment mainly tackles the application of the design patterns you studied during the course, so you are supposed to give time for the design. You should of course use MVC.

- Complete log of the operations done on the game (e.g., user actions, objects intersections, creation or reusing of objects) should be generated. You can use JDK logger, log4J, ...etc. packages for this purpose. Logging all such actions will introduce a delay to the game, so you must use multiple level of logging (e.g. DEBUG, INFO, ...), and classify your logs according its nature.
- You need to support at least 3 levels of difficulties, but you are free to choose any criteria for difficulty (e.g., different speed, multiple clowns, more queues, changing plates colors or sizes, ... etc.).
- You are provided with a custom game engine that supports three types of objects: movable, constant and user-controlled objects. Note : you may have look on the sample game on piazza to see how the engine is included, and how the sample game interacts with the game engine
- Two interfaces define the interaction with the game engine:
  1. World: defines a level of the game and its objects
  2. GameObject: an object of the game, it could be a shape, clown, ... etc.

Integration

You will use the game engine provided with the pdf. The source code and the javadoc of the interfaces are shown below

```java
package eg.edu.alexu.csd.oop.game;
public interface GameObject {
/** setter/getter for X position*/
int getX();
void setX( int x );
/** setter/getter for Y position*/
int getY();
void setY( int y );
/** @return object width */
int getWidth();
/** @return object height */
int getHeight();
/** @return object visible or not */
boolean isVisible();
/** @return object movement frames */
java.awt.image.BufferedImage[] getSpriteImages();
}
```

```java
package eg.edu.alexu.csd.oop.game;
public interface World {
/** @return list of immovable object */
java.util.List<GameObject> getConstantObjects();
/** @return list of moving object */
java.util.List<GameObject> getMovableObjects();
/** @return list of user controlled object */
java.util.List<GameObject> getControlableObjects();
/** @return screen width */
int getWidth();
/** @return screen height */
int getHeight();
/**
* refresh the world state and update locations
* @return false means game over
*/
boolean refresh();
/**
* status bar content
* @return string to be shown at status bar
*/
String getStatus();
/** @return frequency of calling refresh */
int getSpeed();
/** @return frequency of receiving user input */
int getControlSpeed();
}
```

Report

The report should contain the following:

- Describe your design thoroughly.
- Class diagram of your design.
- Sequence diagram showing the typical scenarios of the game.
- Section for each pattern (the required and any other patterns you used) and how you used it in your design, and a class diagram explaining this.
- Any design decisions that you have made should be listed clearly.

Deliverables

- Download the sample eclipse project that uses our game engine. It is available at Piazza resources page.
- You should work in groups of four.
- The implementation for the given interfaces.
- You should pack your game in an executable JAR file .
- This assignment mainly tackles the design issues. Heavy load will be on the good design in addition to the required patterns.
- Develop this assignment in Java.
- You should deliver your source code using your Bitbucket repository.
- Upload your report , jar file , zipped source code to materials folder in your git repository.
- Delivering a copy will be severely penalized for both parties, so delivering nothing is so much better than delivering a copy.

Good Luck