INFORMÁTICA. Curso 20-21

Trabajo individual del bloque II

Completa el proyecto BlueJ que se proporciona como material para la realización de este trabajo, definiendo las clases solicitadas en los diversos apartados del enunciado para resolver el problema planteado.

*La entrega se realizará a través del enlace disponible en Moodle, en el apartado "Trabajos individuales". La fecha límite de entrega es el **17 de diciembre**, jueves.*

***Se entregará un único archivo comprimido (*.7z o *.zip**) con el contenido de la carpeta del proyecto.*

Es importante que previamente a la realización del trabajo planifiques el aprendizaje de los conceptos necesarios para su resolución, que se han trabajado en las clases de teoría, en las clases de problemas y en las prácticas de programación.

*Se recuerda que la elaboración de este trabajo es **personal e individual**. En caso de detectarse trabajos con un excesivo grado de similitud se aplicarán las medidas descritas en el apartado "Copias o plagios" del documento de "Presentación de la asignatura" publicado en Moodle, en el apartado Información.*

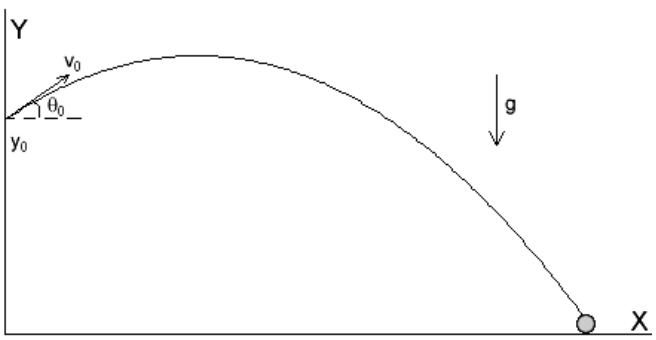
Notas previas

1. Salvo que se indique explícitamente lo contrario, todos los algoritmos solicitados se implementarán como **métodos no estáticos**
2. No se usarán instrucciones condicionales ni iterativas innecesarias
3. No se realizarán tareas no especificadas en el enunciado
4. Todos los recorridos de colecciones finalizarán en el momento adecuado, sin acceder sistemáticamente a todos sus elementos
5. Ningún bucle contendrá ninguna instrucción que provoque su finalización
6. Nunca se volverá a implementar código ya implementado

Enunciado

El objetivo es realizar una simulación de lanzamientos de proyectiles en el **plano real**.

Todos los proyectiles se lanzan desde una determinada altura en el eje de ordenadas, con un ángulo y un valor de velocidad. Mientras están en el aire se ven sometidos a la aceleración de la gravedad hasta que terminan llegando al eje de abscisas, que representa el suelo.



Para poder representar gráficamente esta simulación se proporciona un *package* de Java ya creado, `Graph`, del que **solo se podrán usar los siguientes algoritmos**:

- De la clase **`Recinto`**:

  `Recinto(int width, int height)`
  Constructor. Crea y dibuja el recinto donde luego se podrán dibujar las figuras de los proyectiles. Los parámetros representan las dimensiones del recinto.
  Es imprescindible ejecutarlo antes de dibujar las figuras; en caso contrario se producirá un error de ejecución.
  El origen de coordenadas (0,0) es la esquina inferior izquierda del recinto, y al igual que en el plano real, las coordenadas crecen hacia arriba y hacia la derecha.

  `static void` `wait(int ms)`
  Hace una pausa del número indicado de milisegundos.
  Al tratarse de un algoritmo estático, para poder ejecutarlo desde fuera de esta clase debe precederse del nombre de la clase: `Recinto.wait(…)`

- De la clase **`Circle`**:

  `Circle (int xC, int yC, int rad)`
  Constructor. Crea una figura circular sin dibujarla. Los parámetros representan el centro del círculo y su radio en píxeles.

  `void` `draw()`
  Dibuja el círculo en el recinto.

  `void` `undraw()`
  Borra el círculo del recinto.

  `void` `setCenter(int xC, int yC)`
  Cambia la posición del círculo. Los parámetros representan la nueva posición.

  `void` `setRadius(int rad)`
  Cambia el radio del círculo. El parámetro representa el nuevo valor de radio.

Para tener una referencia de cómo emplear estas utilidades, se proporciona el proyecto `SimulacionDeProyectiles` que contiene un ejemplo de uso de las mismas. Observa que en las clases donde quieras emplear alguna de estas utilidades tendrás que incluir la siguiente cláusula al principio del fichero:

```
import Graph.*;
```

Las clases que se proponen a continuación para resolver el problema se definirán dentro del proyecto `SimulacionDeProyectiles` pero fuera del *package* `Graph`; es decir, junto a la clase dada como ejemplo de uso del entorno. **Deberán respetar escrupulosamente la especificación descrita** en el enunciado.

La representación gráfica de todos los proyectiles será un **círculo de radio 5 píxeles**.

El lapso de tiempo al que se hace referencia en las diferentes clases para actualizar los datos es un número real de segundos.

La velocidad inicial y el ángulo de lanzamiento de todos los proyectiles se tendrán que poder calcular de forma aleatoria, por lo que en alguna de las clases, en la que te parezca más apropiada, habrá que implementar**:**

- **Una función estática que calcule un número real aleatorio entre 0 y un cierto valor máximo**, utilizando la función `random` definida en la clase `Math`, que genera un número real aleatorio entre 0 y 1.

Clase `Posición`

Clase que representa al punto del **plano real** (es decir, con coordenadas reales) en el que está situado el proyectil. Se controlará que la posición del proyectil no llegue a estar por debajo del suelo; es decir, que si se detecta que su valor de ordenada es inferior al radio de su representación gráfica, a la ordenada se le asignará directamente el valor de dicho radio

Contendrá un constructor:

- Crea la posición de lanzamiento del proyectil, que como se ha indicado anteriormente será siempre un punto de la semirrecta positiva del eje `Y`

Contendrá dos métodos:

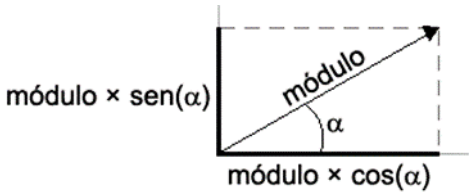
- Comprueba si la posición del proyectil es un punto del "suelo"; es decir, su ordenada no es mayor que el radio de la representación gráfica del proyectil
- Actualiza la posición del proyectil tras verse sometida a una velocidad (clase que se define a continuación) durante un lapso de tiempo. El valor de posición se modifica sumando a la posición actual la velocidad multiplicada por el lapso de tiempo, tanto para las `x` como para las `y`

Clase `Velocidad`

Clase que representa la velocidad a la que se mueve un proyectil, expresada en sus componentes horizontal `vx` y vertical `vy`.

Contendrá un constructor:

- Genera la velocidad de lanzamiento de un proyectil a partir de su módulo (magnitud) y ángulo de lanzamiento



Contendrá un método:

- Actualiza la velocidad del proyectil tras verse sometido a la aceleración de la gravedad (−9.8) durante un lapso de tiempo. El valor de velocidad se modifica sumando a la velocidad actual la aceleración multiplicada por el lapso de tiempo; solo para las `y` pues la gravedad es vertical (en `x` la velocidad no cambia)

Clase `Proyectil`

Clase que representa a un proyectil, con su posición, su velocidad y su figura gráfica, que como se ha indicado anteriormente será un círculo de radio 5 píxeles. También interesa el dato de la mayor altura que ha alcanzado el proyectil

Contendrá dos constructores que diferirán en el modo de obtener la velocidad de lanzamiento del proyectil. Por lo demás, ambos crearán el proyectil en una posición inicial que será siempre un punto de la semirrecta positiva del eje `Y`, y ambos dibujarán también la figura del proyectil en el recinto. Para esto último, observa que aunque **las coordenadas de la posición del proyectil son números reales, las de la figura dentro del recinto son números enteros**, por lo que habrá que realizar la pertinente conversión.

- El primer constructor creará el proyectil con un módulo y ángulo de velocidad de lanzamiento dados
- El segundo creará el proyectil con una velocidad de lanzamiento calculada aleatoriamente: un ángulo aleatorio entre 0 y $\pi/2$, y una magnitud aleatoria entre 0 y 80.0

Contendrá tres métodos:

- Dibuja la figura del proyectil
- Comprueba si el proyectil ha llegado al suelo
- Actualiza los datos del proyectil tras un lapso de tiempo, incluyendo el dato de la mayor altura, de modo que **cuando el proyectil llegue al suelo dejará de moverse**. También tiene que cambiar de sitio la figura en el recinto

Clase `Simulacion`

Clase que representa al conjunto de proyectiles que se van a lanzar y el recinto donde se van a pintar, para lo que es necesario conocer las dimensiones del recinto y el valor del lapso de tiempo en el que se irán actualizando los proyectiles.

Contendrá dos constructores:

- El primero, sin parámetros, servirá para poder hacer la fase de prueba de forma rápida, creando directamente la simulación que se describe a continuación:
  - Un recinto con dimensiones 800×600
  - Un lapso de tiempo de 0.1 segundos
  - Un conjunto con los siguientes 4 proyectiles:

        - Altura inicial: 25; magnitud de velocidad: 40; ángulo: π/10
        - Altura inicial: 50; magnitud de velocidad: 30; ángulo: π/8
        - Altura inicial: 100; magnitud de velocidad: 50; ángulo: π/4
        - Altura inicial: 200; magnitud de velocidad: 20; ángulo: π/3
- El segundo constructor creará la simulación “buena”:
    - A partir del número de proyectiles que contendrá el conjunto, de las dimensiones del recinto y del valor del lapso de tiempo de actualización de los proyectiles, creará el recinto con esas dimensiones, y creará el conjunto con ese número de proyectiles, de manera que las ordenadas de sus posiciones iniciales sean los primeros múltiplos de 20 (20, 40, 60…) y sus velocidades sean aleatorias

Contendrá ocho métodos:

- Actualiza todos los proyectiles del conjunto
- Comprueba si todos los proyectiles han llegado al suelo
- Implementa la simulación del lanzamiento de los proyectiles: se va actualizando el conjunto para el lapso de tiempo establecido hasta que todos los proyectiles llegan al suelo
- Calcula la mayor altura alcanzada por un proyectil del conjunto
- Calcula la mayor distancia horizontal alcanzada por un proyectil del conjunto
- Calcula la menor distancia horizontal alcanzada por un proyectil del conjunto
- Calcula la posición promedio de los proyectiles del conjunto
- Calcula cuántos proyectiles han llegado como mínimo a la mitad de la anchura del recinto

Ejercicio final

La realización del enunciado propuesto hasta el momento se valorará sobre **9 puntos**. Para poder optar a la nota máxima habrá que conseguir que el método que implementa la simulación del lanzamiento, detecte si ésta ya haya finalizado, en cuyo caso volverá a realizar un nuevo lanzamiento; es decir, recolocará los proyectiles en su posición inicial, les asignará de nuevo velocidades aleatorias, y después procederá al lanzamiento.

Para que en dicho método sea posible saber si la simulación ha finalizado o no, convendría añadir una propiedad `boolean` a la clase `Simulacion`. Y para dejar el recinto “limpio” y poder modificar los proyectiles del modo descrito, convendría añadir algunos métodos adicionales a la clase `Proyectil`.

La calificación de la actividad dependerá del grado de adquisición de las siguientes **competencias**:

1. **Escritura correcta del código Java**: sangrado y comentarios explicativos
2. **Aplicación correcta de los conceptos de programación**: nombres adecuados; datos del tipo adecuado; cabeceras de métodos bien diseñadas; instrucciones adecuadas, lógicas y coherentes…
3. **Toma de decisiones adecuadas** en la especificación de los procesos
4. **Consecución del objetivo**: resolución correcta del problema planteado
5. **Dedicación e interés demostrados** en el desarrollo de la actividad