Ing. de Sistemas e Informática
Estructuras de Datos y Algoritmos
Jorge Mario Londoño P.

Taller 1

Tipos de datos abstractos (ADT) y estructuras básicas

Introducción

Una biblioteca de funciones encapsula tipos de datos abstractos y algoritmos de uso frecuente en el desarrollo de software.

Los autores de el texto guía ofrecen un API para facilitar operaciones frecuentemente utilizadas. La biblioteca de funciones se encuentra disponible en la siguiente dirección:

http://algs4.cs.princeton.edu/code/

Allí están disponibles el código fuente, la documentación y archivos ejemplos para los distintos algoritmos. La biblioteca entera está disponible como un archivo JAR (algs4.jar), el cual se debe incluir en el CLASSPATH del proyecto (ver instrucciones en la misma página).

Para efectos de esta práctica nos interesan 5 clases incluidas en esta biblioteca:

`StdIn` : Funciones para lectura de datos por consola

`StdOut` : Funciones para escritura de datos en consola

`In` : Funciones para leer datos desde archivos o URLs

`Out` : Funciones para escribir datos en archivos

`StdDraw` : Funciones para hacer gráficas

`StdRandom` : Funciones para generar números aleatorios

`StdStats` : Cálculos estadísticos básicos

Otra biblioteca de uso frecuente está incluida en el API de Java es la clase `Arrays`, que nos permite hacer varias cosas utilizando arreglos de datos: Ordenarlos, convertirlos a `String`, hacer búsquedas, etc. La clase está documentada aquí: `java.util.Arrays`.

Ejercicio a desarrollar

Se desea implementar una simulación (simplificada) del movimiento de las `BolaDeBillar` sobre una mesa. Para tal efecto se requiere implementar el ADT `BolaDeBillar` que representa las bolas de billar por medio de sus coordenadas en el plano (x,y) y su vector velocidad (vx,vy). Dentro del API de la `BolaDeBillar` se necesitan las siguientes operaciones:

`draw` : Dibuja la bola de billar en sus coordenadas actuales.

`move` : Ajusta la posición según un parámetro delta_t que representa el tiempo desde el último movimiento. (Se asume velocidad constante).

`collision` : Detecta si ocurre una colisión con uno de los bordes de la mesa y ajusta el vector velocidad así:

- Si golpea contra los extremos izquierdo o derecho, la velocidad en x cambia de signo. vy sigue igual.
- Si golpea contra los extremos inferior o superior, la velocidad en y cambia de signo y la vx sigue igual.
- `collision` toma como argumento un rectángulo (RectHV) definido más abajo para especificar las dimensiones de la mesa.

Para representar la mesa de billar (y como clase principal de la simulación) se define el ADT `MesaBillar`, el cual realiza las siguientes funciones:

- Define un Bag que contiene las `BolasDeBillar`.
- Define un rectángulo horizontal-vertical (RectHV) que representa el área de la mesa (y en particular las coordenadas de sus bordes para detectar las colisiones).
- El constructor acepta un parámetro N y crea N objetos `BolaDeBillar` con posiciones y velocidades aleatorias.
- La única operación del API es runSimulation la cual consta de un ciclo para simular el movimiento por T unidades de tiempo. Las T unidades de tiempo se dividen en intervalos delta_t y por cada intervalo se debe: Determinar si tuvo colisión, hacer el movimiento correspondiente a un delta_t y dibujar las `BolaDeBillar` en su nueva posición.

El método main de `MesaBillar` inicializa la simulación con N BolaDeBillar e invoca el método `runSimulation`.

Entregables:

Clases implementado los ADT definidos: `BolaDeBillar y MesaBillar.`

Métodos main: En `BolaDeBillar` hace pruebas unitarias de la clase. El de `MesaBillar` es el main principal del programa.

Enviar solo el código fuente como un archivo comprimido (se aceptan .zip, .rar, .7z, .tgz). No incluir dentro del comprimido la biblioteca algs4.jar.

Se puede realizar el trabajo en equipos de máximo 2 personas.

Seguir el estándar de nombres: Practica1-<NombreApellido1>-<NombreApellido2>.zip.