Departamento de Informática
Universidad Carlos III de Madrid

Grado en Ingeniería Informática

Programación
Examen Final

Diciembre 2019

Lee atentamente estas instrucciones, en caso de no seguirlas se aplicará una penalización mínima de 1 punto

- El tiempo para realizar el examen es de **3 horas**
- No respondas en esta hoja, usa las hojas en blanco.
- Esta hoja no se debe entregar, se puede anotar en ella lo que se considere oportuno.
- Rellena el examen con bolígrafo, no se recogerán exámenes escritos a lápiz ni con bolígrafo rojo.
- Usar una hoja para cada clase. Si no se resuelven los apartados en orden se debe usar una cara para cada apartado y los apartados deben estar en el orden correcto al final.
- No olvides poner el nombre y grupo en todas las páginas.
- Se puede utilizar cualquier tipo de material impreso. Los dispositivos electrónicos están prohibidos.
- No se puede abandonar el aula a no ser que se entregue el examen.
- No está permitido:
  - Usar funciones o librerías no vistas durante el curso.
  - Usar `break` o `continue`.
- Los únicos métodos de lista permitidos son: append, insert y remove
- Lee el examen entero antes de empezar.

Sistema de gestión de almacenes

Empresas de distribución como Amazon están experimentando con el uso de drones para entregar mercancías a sus clientes. Esto reducirá los costes de entrega y el tiempo necesario para hacerlo. El objetivo de este ejercicio es simular un sistema de gestión de almacenes, donde se utilizan drones, de acuerdo con la siguiente descripción:

- Cada almacén tiene un nombre, una serie de drones, entrega paquetes en un vecindario determinado y es capaz de recibir pedidos.
- Un dron tiene un id, una carga actual (en el rango de 0 a 100), un estado (disponible u ocupado) y se puede utilizar para entregar un pedido (inicialmente no tienen ningún pedido).
- Un pedido tiene una prioridad, un id y la dirección donde tiene que ser entregado.
- El mapa del área tiene un nombre del vecindario y es una matriz donde cada elemento es una cadena que representa el identificador de una casa o 's' para las posiciones donde no hay casas (ver ejemplo más abajo)
- Para cada pedido en el almacén, se selecciona un dron para servirlo. Más abajo están los detalles sobre cómo se seleccionan los drones.

Teniendo en cuenta la descripción anterior, se pide:

1. (1pt) Definir las clases necesarias y sus métodos `init` teniendo en cuenta que se deben crear propiedades para los atributos que las necesiten. Crear propiedades no necesarias será penalizado.
2. (0,5pt) Crear el método `crearDrones(self,Num)` que devuelve una lista de `Num` drones con ids consecutivos `dron1`, `dron2`, ..., `dronNum`. Indica la clase a la que pertenece este método.
3. (2pt) Crear el método `crearMapaArea(self,whId,rows,cols,NumH)` que, dado el nombre del almacén (`whId`), la dimensión del área en términos de número de filas (`rows`) y número de columnas (`cols`), y el número de casas en el área (`NumH`), devuelve una matriz `rows`×`cols` representada como una lista de listas,

que almacena cadenas de texto para representar los elementos en el mapa. La cadena "s" representará calles sin casas de cliente en ellas; la cadena `whId` representará la posición del almacén; y las cadenas `"h1"` a `"hNumH"` representarán los identificadores de las casas. La posición de cada elemento del mapa se seleccionará aleatoriamente. Hay que asegurar que todas las casas se crean y las direcciones de las casas no se repiten. Indica la clase a la que pertenece este método.

Ejemplo de un mapa de a 4×5 con 10 casas para el almacén `wh0`:

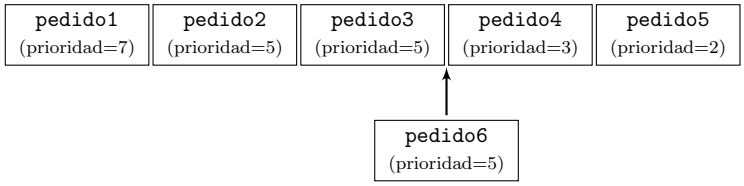
s	h10	s	h7	h8
s	s	s	h9	wh0
s	s	h3	h2	h4
h5	s	s	h1	h6

4. (2pt) Crear el método `calcularDistancia(self,whId,houseId)` que dado el identificador de almacén (`whId`) y un identificador de casa (`houseId`), devuelve la distancia euclídea (o la distancia en línea recta) entre los dos. La distancia euclídea se calcula utilizando la siguiente fórmula: $d = \sqrt{(r_o - r_w)^2 + (c_o - c_w)^2}$, donde r_o y c_o hacen referencia a la fila y la columna donde se encuentra la dirección de la casa, y r_w y c_w se refieren a la fila y la columna donde se encuentra el almacén. La raíz cuadrada de un número x se puede calcular utilizando $x * *0.5$ Indica la clase a la que pertenece este método.

   Ejemplos: la distancia euclídea entre `wh0` y `h5` es $d = \sqrt{(3-1)^2 + (0-4)^2}$; la distancia entre `wh0` y `h3` es $d = \sqrt{(2-1)^2 + (2-4)^2}$.

5. (1pt) Crea el método `insertarPedido(self,pedido)` que dado un pedido, lo inserta en la lista de pedidos según su valor de prioridad (pedidos de prioridad más alta primero). Si hay más de un pedido con los mismos valores de prioridad, se insertará al final de los pedidos con la misma prioridad. Este método no devuelve nada. Indica la clase a la que pertenece este método..

   Ejemplo: si hay que insertar el pedido `pedido06` que tiene prioridad 5:



6. (2pt) Crear el método `elegirDron(self,direccion)` que dada una `direccion` de pedido, devuelve la posición que ocupa el dron que lo va a entregar, en la lista de drones del almacén y la distancia desde el almacén a `direccion`. El dron elegido será el primero disponible cuya carga sea al menos dos veces la distancia desde el almacén hasta la `direccion`. Si no se encuentra un dron que cumpla estas características, el que tenga la carga más alta se recargará inmediatamente por completo y se seleccionará. Indica la clase a la que pertenece este método.

7. (1pt) Crear el método `procesarPedido(self,pedido)` que inserta el `pedido` en la lista de pedidos del almacén, elige el dron más adecuado para procesarlo y actualiza los atributos pertinentes del dron elegido. Este método no devuelve nada. Indica la clase a la que pertenece este método.

8. (0.5pt) En un programa principal, pregunta al gerente del almacén el número de drones con los que cuenta el almacén, el número de filas y el número de columnas que cubre el área, y el número de casas en el área. A continuación, crea un almacén; para dicho almacén, crea cinco pedidos para procesar con ids consecutivos `pedido1`, `pedido2`, ..., `pedido5`.