PRÁCTICA 4

Modelado y Programación

Realiza las siguientes actividades

1. Fila bancaria

Simula en python una fila bancaria.
Deberás simular el proceso de cómo llegan clientes a un banco a solicitar transacciones de diversos tipos, algunos son pagos, retiros de dinero, etc.
Cada cliente tarda entre 1 minuto y 15 minutos en ser atendido, pero tu programa convertirá los minutos en segundos para evitar que sea demasiado tardada la ejecución.

Supón que tienes 3 ventanillas en el banco, el banco da servicio de 12:00 a 13:30 horas y cada 2 minutos llega un nuevo cliente.

Ejemplo de Ejecución:

12:00 -> persona1 pasa a caja 1, transacción: 3 minutos
12:01 -> persona1 en caja 1
12:02 -> persona1 en caja 1
persona2 pasa a caja 2, transacción: 10 minutos
12:03 -> persona2 en caja 2
persona1 sale de caja1
12:04 -> persona2 en caja 2
persona3 pasa a caja1, transacción 8 minutos
12:05 -> persona2 en caja 2
persona3 en caja 1
12:06 -> persona2 en caja 2
persona3 en caja 1
persona 4 pasa a caja3, transacción 11 minutos
12:07 -> persona2 en caja 2
persona3 en caja 1
persona4 en caja 3
12:08 -> persona2 en caja 2
persona3 en caja 1
persona4 en caja 3
persona 5 en espera, transacción: 5 minutos
12:09 -> persona2 en caja 2
persona3 en caja 1
persona4 en caja 3
persona 5 en espera, transacción: 5 minutos
12:10 -> 12:09 -> persona2 en caja 2
persona3 en caja 1
persona4 en caja 3
persona 5 en espera, transacción: 5 minutos

persona 6 en espera, transacción: 1 minuto
12:11 -> persona2 en caja 2
persona3 en caja 1
persona4 en caja 3
persona 5 en espera, transacción: 5 minutos
persona 6 en espera, transacción: 1 minuto
12:12 -> persona2 sale de caja 2
persona3 en caja 1
persona4 en caja 3
persona 5 pasa a caja 2, transacción: 5 minutos
persona 6 en espera, transacción: 1 minuto
etc…

Una vez que son las 13:30 no debe de formarse nadie más en la fila, pero sí debes terminar de atender a la gente que se quedó en la fila e imprimir el estado de la fila cada minuto hasta que ya no haya clientes.

2. Herencia

Crea la clase llamada Fruta, una fruta debe tener tamaño (grande, mediana, pequeña) y color. (puede ser el color habitual de la fruta y el color de cuando aún no está madura).
Crea las clases Fresa, Mango, Sandía y Toronja que extiendan a Fruta; estas clases deben agregar las propiedades: sabor (dulce, ácido o amargo) y la propiedad conCascara que será una variable booleana que valdrá true en caso de que la fruta posea una cáscara no comestible, como el caso de la sandía en nuestra lista de frutas.
Deberás crear una lista de tamaño 6 con frutas del mismo tipo con sabores, colores y tamaños al azar. Cada lista de frutas se puede identificar como *canasta.*
Tendrás que implementar métodos que permitan realizar las siguientes operaciones:

- *contiene(self, fruta)*

Esta función regresa true cuando una fruta está dentro de una canasta

- *revuelve(self, canasta)*

Esta función regresa una canasta de longitud 12 que contendrá frutas de dos tipos distintos

- *limpia(self)*

Esta función recibe una canasta de frutas que pueden ser de tamaños y sabores distintos, pero sólo se queda con las frutas que son de sabor dulce y tamaños grande o mediano eliminando las que no lo son.

- *imprimeCanasta(self)*

Este método recibe una canasta de frutas e imprimirá en pantalla cada fruta. Ejemplo:
Canasta de fresas:
fresa roja mediana dulce - fresa roja pequeña ácida - fresa roja mediana dulce - fresa verde grande ácida - fresa roja grande amarga - fresa roja pequeña dulce

Fecha de entrega: 18 de Septiembre de 2016