Informe Trabajo Práctico

Estructura de Datos

Profesores:
Maceira, Carlos
Abosso, Franco

Grupo 6 comisión C:
Dartiguelongue, Gonzalo
Gutierrez Zaldivar, Pedro
Bertao, Jonathan

Objetivo:
El objetivo principal de este trabajo práctico es simular el comportamiento de una central de red, que puede tener teléfonos incluidos en ella, mediante el uso de python como programación orientada a objetos (POO). Los teléfonos deben tener sus respectivas funcionalidades, siempre con el objetivo de orientarlo hacia como es en la realidad. La idea principal es que se puedan realizar llamadas y envíos de mensajes de unos teléfonos a otros y que estos se almacenen en archivos. Finalmente, con los distintos datos de llamadas y mensajes, se debe realizar un análisis de la información con el objetivo de obtener conclusiones relevantes que puedan ser utilizadas como herramienta para planificar estrategias,ofrecer promociones, o tomar una decisión de mejora en uno o más aspectos. Para lo cual se debe utilizar la librería matplotlib.

Explicación del tp:
Para poder cumplir con nuestros objetivos creamos distintas clases. Las clases son:
Teléfono, Central, App, Appstore, Configuración, Mail, Llamada, Mensajería, Paint, Nodo y Lista enlazada. Cada clase tiene sus métodos y sus atributos propios. Cada método tiene incorporado un “docstring” que explica y ayuda a comprender su funcionalidades. Algunas de las clases como Appstore, Configuración, Mail, Llamada, Mensajería y Paint fueron creadas con la idea de modularizar y son instanciadas como atributos de Teléfono. De esta forma, desde cada teléfono se pueden llamar los métodos de las clases usadas como atributos, simulando la interacción del teléfono con las aplicaciones.
La clase Central es instanciada una vez nada más ya que para este proyecto estamos trabajando con una central nada mas. A esta instancia están asociados todos los objetos Teléfono. Esta es en la que se registran los teléfonos instanciados y que administra, realiza y registra todas las interacciones de llamadas y mensajes entre los teléfonos, luego también puede dar un informe detallado sobre las llamadas realizadas y mensajes enviados y recibidos.
La clase Configuración es la que administra el estado de la red móvil, los datos y el nombre de cada teléfono.
La clase Llamada es la que registra dentro de cada teléfono las llamadas recibidas y realizadas de dicho teléfono.
La clase Mensajería es la que registra dentro de cada teléfono los mensajes mandados y recibidos por dicho teléfono.
La clase Mail es la que almacena y permite ver la bandeja de entrada dentro de cada teléfono.
La clase App es instanciada dentro de la clase Appstore para poder tener el objeto de cada app con todas sus especificaciones como atributos.
La clase Appstore es la que tiene almacenada todas las apps disponibles y las que un teléfono tiene descargadas. También permite al teléfono descargar nuevas apps y eliminar las que ya tiene descargadas.
La clase Paint es instanciada como atributo de teléfono y esta consta de un método que utiliza la librería pygame y que te permite dibujar en una pantalla.

La clase Telefono es la clase que tiene como atributo la mayoría de las demás clases. Es la que se instancia cada vez que se crea un telefono. Esta clase cuenta con numerables métodos que sirven para encender y apagar el teléfono, su red móvil o sus datos móviles, también para bloquear y desbloquear el teléfono, y para descargar y eliminar apps entre otros.
Para todos los métodos que creamos tuvimos que validar los datos que reciben y el estado del objeto que los usaba, por ejemplo si la red móvil del teléfono estaba activa para recibir una llamada.
Para el análisis de datos creamos distintas funciones que nos permiten, utilizando la librería matplotlib, hacer gráficos de tortas y de barras. El gráfico de barras toma todos los teléfonos creados y en base a la cantidad de teléfonos por sistema operativo crea las barras. Donde en mi X van a aparecer los distintos sistemas operativos y en mi Y figuran las cantidades de teléfono que lo tienen. En el gráfico de barra por otro lado figuran la cantidad de llamadas con cierta duración. Donde tomaremos en minutos (el número entero) las duraciones, siempre aproximando para arriba. Esto se representa en porcentajes dentro del gráfico de barra.
Finalmente, en el main pusimos a prueba todo lo que creamos. En el main instanciamos la central y varios teléfonos. Después simulamos la modificación exitosa y fallida de la configuración de un teléfono. Luego simulamos interacciones exitosas y fallidas entre los teléfonos a través de la central y mostramos por pantalla los historiales y registros de llamadas y mensajes. También simulamos la descarga y eliminación exitosa y fallida de aplicaciones. Finalmente abrimos la app paint y por último la exposición por pantalla de los gráficos con el fin del análisis de datos.

Estructuras de datos:
A continuación proporcionamos las estructuras de datos utilizadas en distintos casos y su explicación de porqué fueron seleccionadas:
<u>En Central:</u>

Usamos dos diccionarios distintos. Ambos almacenan como key un número de teléfono, pero uno tiene como value un booleano (“telefono_modo_avion”) y el otro (“telefonos_registrados”) tiene como value un objeto de tipo telefónico correspondiente al número de teléfono que es key. Ambos diccionarios contienen todos los teléfonos registrados en un cierto momento. Hicimos uno con value booleano para almacenar si el teléfono está o no en modo avión, para validar fácilmente con esto que puede o no puede hacer ese teléfono. Utilizamos diccionarios porque como el número de teléfono es único, se nos ocurrió usarlo como key ya que las keys son únicas en los diccionarios. Además, esto permite un fácil manejo de los datos para búsqueda y demás.
<u>En Telefono:</u>

Para los contactos usamos conjuntos, ya que no pueden tener elementos repetidos, así como no se puede tener un mismo contacto.
<u>En Appstore:</u>

Dos diccionarios, uno para todas las apps que sirve para tener todas las que son posibles de descargar, es decir, que existen. Tiene como value el nombre de la app y como valor el objeto del tipo de app. El otro diccionario es para las apps descargadas que tiene ese teléfono. Al descargarse una app se agrega a este diccionario como clave el nombre de la app, y como value el tipo de clase de la app. Al trabajar con misma lógica en ambos diccionarios se

hace simple el manejo de aplicaciones, descargas y eliminaciones. La asociación key-value nos facilita el manejo de datos, y además con la key guardamos los nombres de las apps que tiene sentido que no se puedan repetir.

En Mail:

Una lista de listas que representa una bandeja de entrada de mails, la cual la creamos manualmente. La lista contiene listas con un mensaje, una fecha y una cadena que representa si el mensaje fue leído o no leído. De esta forma se hace muy fácil mostrar por pantalla los mails ordenados según no_leidos-leidos y según fecha que es lo que se pedía.

En Llamada:

Utilizamos lista enlazada para guardar el historial de llamadas. Sirve para que se guarden en orden de realización, es decir para un acceso secuencial más eficiente. También es un uso de memoria optimizado. Esta última razón a los fines prácticos de nuestro Tp no cambian en mucho pero en clase comentaron que cuando se tienen códigos muy grandes es muy importante la optimización y nombraron las listas enlazadas en estos casos como una de las posibles soluciones para ello, es por eso que lo aplicamos.

En Mensajeria:

Usamos la librería “deque” que nos permite trabajar con pilas y colas a nuestra conveniencia. La usamos para la bandeja de entrada de sms y para el historial de sms enviados. Sirve para ir agregando mensajes en el último lugar a medida que llegan, pero si hay que ver el primero, sale el primero en llegar y no el último.

README:

Para que todo funcione correctamente debe tener descargado las librerías matplotlib y pygame.
Para instalar matplotlib corra en la central:

```
python -m pip install -U pip
python -m pip install -U matplotlib
```

Para instalar pygame corra en la central:

```
pip install pygame
```

Para crear un teléfono, se debe utilizar un try-except como verán en el código “Main” para que se valide que no haya teléfonos con mismo numero de telefono, ya que el numero es unico.
La realización de llamadas y envíos de mensaje de un teléfono a otro se hace desde la central, por lo que para poder simular estas acciones se tienen que utilizar los métodos de la central, que a su vez va a hacer que se utilicen los métodos correspondientes de cada teléfono.
Para descargar aplicaciones, ver mails según orden o fecha, correr la aplicación paint, agregar contactos, ver historiales de llamadas/mensajes, y para modificar algún atributo del teléfono se hace con métodos directamente desde el teléfono.
Una consideración a tener en cuenta es que pusimos manualmente una fecha en el main para cuando se realizan llamadas, y estas se guardan en el archivo “llamadas.csv” con sus respectivos estados que pueden ser ocupado o conectado, pero luego si vuelve a correr el main, al tener en el archivo las llamadas de antes, los estados de todas las llamadas van a ser ocupado ya que se intenta de hacer una llamada con la misma fecha de las que ya están en el archivo. Nosotros modelizamos para que se pueda realizar una llamada, la nueva llamada debe

ser mas tarde que la ultima llamada mas el tiempo de duración de ella. Si se llama a un teléfono en un horario en el que ya estaba en llamada, el estado de esa llamada será ocupado. Si un telefono esta en llamada, ni siquiera puede realizar una llamada.