Laboratorio 3

Objetivos:

- Analizar un contexto a fin de modelar el diagrama de clases correspondiente.
- Codificar el diagrama de clases en el lenguaje de programación Java.
- Aplicar buenas prácticas de programación y el uso de estándares de codificación.
- Utilizar javaDoc para documentar el código y facilitar la generación automática del API de la clase.

Contexto

Un **Chip Prepago** permite la operación del teléfono celular al acceder a los servicios de la red celular que provee el operador (empresa de servicios de telecomunicación). El chip se asocia con un número de teléfono celular cuando se crea un objeto. Pero la información adicional solamente se registra cuando se procede a su activación.

Por cada minuto de llamada, se consumen 30 colones de saldo. Los montos de mensajería y llamada internacional los define el estudiante. Además, todas los chips prepago comparten el mismo código de país, que en este caso se considera una constante con el valor 506.

Un chip prepago presenta los siguientes comportamientos:

- activar(...): Permite la asignación de un dueño al chip, además existe un bono inicial de 1000 colones. También se indica la cantidad de megabytes para navegar. El chip prepago debe estar activado para realizar cualquier de las operaciones que se detallan abajo.

- Consultar saldo disponible( ): regresa el saldo actual asociado al chip prepago.

- Consultar actividad de salida(número de otro chip prepago): regresa el listado de mensajes enviados y llamadas realizadas al número indicado.

- recargar: se indica un monto y se aumenta el saldo actual con el monto dado, además se regresa el nuevo saldo de la línea. No se aceptan montos negativos.

- sálvame: si el saldo es cero, se recarga automáticamente 100 colones. Solamente es posible ejecutar esta opción un máximo de 3 veces. Debe retornar un mensaje de éxito o fallo en la transferencia.

- llamar: se indica el número de minutos que duró la **llamada** y el número de teléfono destino, así como la fecha y hora (consultados al sistema operativo). Se regresa el saldo actual del chip prepago, luego de haber aplicado los cargos. Es importante mencionar que cuando es una llamada de emergencia al 911, la misma es gratuita. Recuerde que los saldos nunca son negativos.

- Consultar historial de llamadas: retorna un reporte con todas las llamadas realizadas hasta la fecha (duración, número de teléfono destino, fecha y hora)

- Consultar historial de mensajes: retorna un reporte con todos los enviados hasta la fecha (el mensaje original, el número de teléfono destino, fecha y hora)

- Transferir: permite transferir el monto indicado a otro chip prepago. El monto a transferir no puede exceder el saldo disponible. Se aplica un cobro por transferencia de 5 colones.

- Enviar SMS: permite enviar un mensaje de texto a otro chip prepago. El **mensaje** no puede exceder los 128 caracteres y también se debe indicar el número de la línea prepago que recibe el mensaje. El costo del mensaje es de 20 colones. Debe retornar un mensaje de éxito o fallo.

- Ver mensajes recibidos: Permite consultar los mensajes recibidos.

- Navegar: es posible que el usuario visite alguna página web (considere que la página que se quiere visitar es siempre accesible). Para ello debe indicar la URL de la página que visita y se debe calcular aleatoriamente dentro del método una cantidad de kilobytes consumidos, esta cantidad debe afectar la cantidad disponible de la línea prepago. Se retorna la cantidad de megabytes disponibles. El rango de kilobytes es de 1 a 8.

- Consultar cantidad de líneas prepago: debe retornar la cantidad de objetos creados a partir de la clase.

- imprimirReporte(): retorna una cadena con el reporte de llamadas y mensajes realizados en el mes actual. Si el reporte está vació, regresa una cadena “No hay registros de actividad”

- imprimirReporte(mes): retorna una cadena con el reporte de llamadas y mensajes realizados en el mes del año actual indicado. Si el mes es incorrecto regresa el mensaje “Mes no válido”

Indicaciones importantes:

1. Considere crear un método privado que permita verificar si existe saldo disponible para realizar exitosamente las operaciones de envío de mensajes, llamadas o navegación.

2. Las operaciones de envío de mensajes, llamadas o navegación que se realizan exitosamente, deben dejar registros de cada actividad en sus correspondientes arreglos. Los arreglos a utilizar son de tamaño estático, declárase de tamaño 10. Recuerde considerar el tamaño del arreglo al agregar nuevos elementos.

3. Debe utilizar encapsulamiento estricto para los miembros de la clase, a excepción de la interfaz pública de servicios.

4. Utilice arreglos estáticos para almacenar la lista de llamadas y mensajes. Es altamente recomendable que defina estos elementos mediantes dos clases adicionales (Llamada y Mensaje).

5. Todas las clases de este contexto deben ubicarse en el paquete logicadenegocios.

6. Para este contexto puede prescindir de los métodos accesores de tipo *setXXX*().

7. Debe incluir documentación interna mediante tags de javaDoc, excepto a los métodos accesores.

8. Sobrecargar los métodos llamar y enviar mensaje para que reciban además un código de país. Determine al menos 5 países con un coste de servicio diferente.

9. Los métodos enviar mensaje, realizar llamada y transferir monto, implica que estos deben recibir como parámetro un objeto de tipo ChipPrepago para efectuar las operaciones.

Por hacer:

1. Realizar el diagrama de clases de bajo nivel (con detalles de implementación) para la clase ChipPrepago descrita anteriormente. Utilice la herramienta StarUML o LucidChart.

2. Escribir en Java las clases necesarias, incluya todos los atributos necesarios e implemente los métodos solicitados. Utilizar exclusivamente BlueJ.

3. Crear el paquete “aplicacion” y dentro del mismo crear la clase AplChipPrepago. En EL MéTODO MAIN() realice la creación de objetos y ejecute **todas** las pruebas necesarias para garantizar que sus métodos se comportan de acuerdo a las indicaciones.

¿Qúe debo entregar?

- PDF del diagrama de clases.
- .zip del proyecto realizado en BlueJ
- Nombre del archivo: PRY_LAB3_Nombre_Apellido.zip

RECUERDE MANTENER LA CONSISTENCIA ENTRE EL MODELO Y LA CODIFICACIÓN