

Prueba Especial, Programación II

Prof. Rodrigo Olivares
Ayud. Juan Carlos Rojas
Septiembre 07, 2016

Instrucciones:

- El puntaje máximo de la prueba especial es 100%, siendo el 60% el mínimo requerido para aprobar.
- Responda cada pregunta en el lugar indicado. No se aceptarán correcciones de pruebas respondidas con lápiz grafito.
- El tiempo máximo de la evaluación es de 90 minutos.
- La prueba especial es **individual**. Cualquier intento de copia, será sancionado con nota **1,0**.

1. *40pts.* De las siguientes afirmaciones, encierre en un círculo la o las alternativas correctas (*4pts c/u*).

- | | |
|---|--|
| i. La orientación a objeto es: | vi. Para una hebra o hilo se debe: |
| (a) Un paradigma de programación procedural. | (a) Iniciar con el método run. |
| (b) Un paradigma de programación estructurado. | (b) Iniciar con el método start. |
| (c) Una herramienta de programación. | (c) Sobrecribir el método run. |
| (d) Un lenguaje de programación. | (d) Sobrecribir el método start. |
| (e) Ninguna de las anteriores. | (e) Dormir (sleep) la hebra. |
| ii. El principio de ocultamiento: | vii. En el ciclo de vida de una hebra, el estado: |
| (a) Es una técnica que protege el estado de una entidad. | (a) New crea la hebra. |
| (b) Es útil en enfoques procedurales. | (b) Runnable ejecuta siempre la hebra. |
| (c) En Java, se logra con los modificadores de acceso. | (c) Blocked se ejecuta, sin importar estados internos. |
| (d) Es encapsular el conocimiento de una entidad. | (d) Dead es invocado generalmente por el método sleep. |
| (e) Ninguna de las anteriores. | (e) Yield, verifica el desempeño del estado Runnable. |
| iii. Una interface: | viii. Los bloqueos de recursos compartidos se consiguen: |
| (a) Tiene al menos un método implementado. | (a) Package, bloqueando los accesos a las clases internas. |
| (b) Tiene todos sus métodos abstractos. | (b) Clase, bloqueando métodos y atributos de la clase. |
| (c) Es factible de ser implementada. | (c) Atributo, declarándolos como static. |
| (d) Es factible de ser extendida. | (d) Objeto, declarando los métodos como synchronized. |
| (e) Ninguna de las anteriores. | (e) Ninguna de las anteriores |
| vi. La herencia múltiple: | ix. Referente a JFrame: |
| (a) Permite heredar diverso compartimiento. | (a) Habitualmente se usa para crear la ventana principal. |
| (b) Apoya el principio ocultamiento. | (b) getContentPane() obtiene el panel principal. |
| (c) Apoya el principio de encapsulamiento. | (c) setAdd() permite agregar componentes al panel. |
| (d) En Java se desarrolla implementado clases abstractas. | (d) setSize() permite dimensionar la ventana. |
| (e) En Java se desarrolla implementado interfaces. | (e) Ninguna de las anteriores |
| v. Un thread: | x. Para realizar acciones desde un botón Se requiere: |
| (a) Es un flujo de un proceso en memoria. | (a) Crear una clase que implemente un ActionEvent. |
| (b) Es un proceso que se ejecuta en memoria. | (b) Crear una clase que implemente un ActionListener. |
| (c) Puede ser creado como clase en Java. | (c) Re-escribir el método actionPerformed(ActionEvent). |
| (d) Puede ser instanciado como atributo. | (d) Re-escribir el método actionPerformed(ActionEvent). |
| (e) Ninguna de las anteriores. | (e) Agregar la instancia de la clase oyente, al botón. |

2. *60pts*. Desarrolle una aplicación en JAVA, con interfaz de usuario que permita mostrar en algún componente de salida, la información bancaria de abono y giro de un cliente, simulado por procesos concurrentes. Para ellos se solicita lo siguiente:
- (a) Construir interfaz de usuario en JAVA que contenga como mínimo lo que se muestra en la Figura 1.
 - i. El botón “Detener” permiten pausar el proceso en cuestión, cambiando su etiqueta a “Procesar”.
 - ii. El botón “Procesar” permite levantar el proceso en cuestión.
- Utilice los métodos *suspend* y *resume* para pausar y reiniciar las hebras, respectivamente (*10 pts*).
- (b) Construir los procesos concurrentes:
 - i. ProcesoBancoAbono: Permite registrar abonos aleatorios (entre \$1 y \$10.000 CLP) sin límite máximo (*15 pts*).
 - ii. ProcesoBancoGiro: Permite registrar giros aleatorios (entre \$1 y \$10.000 CLP) sin límite mínimo (*15 pts*).
 - (c) Luego de construir los procesos concurrentes, debe desarrollar la clase de recurso compartido (*15 pts*)
 - (d) Por último, desarrolle la clase principal de la aplicación (*5 pts*).

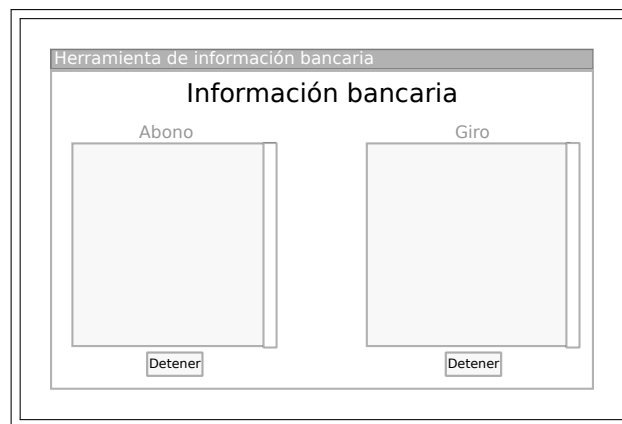


Figura 1: Interfaz de usuario