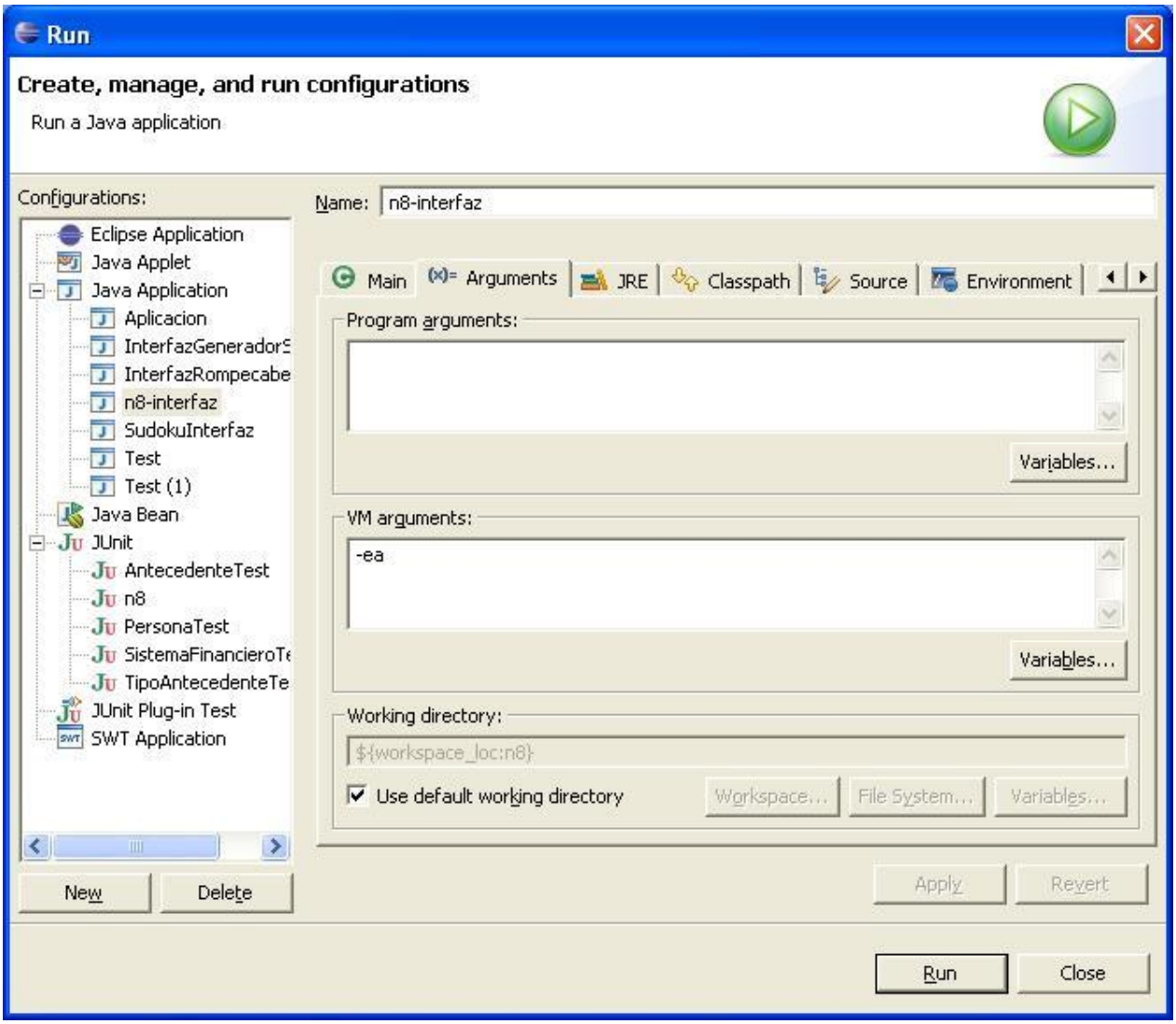
Objetivos

El objetivo de este ejercicio es que el estudiante comprenda y adquiera práctica en:

- El desarrollo de aplicaciones siguiendo un proceso incremental
- El desarrollo de pruebas unitarias en junit para las clases del ejercicio.
- La creación y captura de distintos tipos de excepciones para informar al usuario de manera conveniente cualquier problema detectado.
- La construcción de métodos para hacer persistir la información del modelo del mundo por medio del mecanismo de serialización.
- La construcción de métodos para exportar información hacia archivos secuenciales de texto, utilizando las clases que provee java para manejo de archivos y manipulación de cadenas de caracteres.
- La utilización del control JComboBox del ambiente gráfico de java como mecanismo para manejar listas desplegables en la interfaz de usuario.

Preparación

1. Localice el archivo n8_demo.zip, decomprímalo y ejecute el programa .exe que muestra una ejecución del programa. Estudie el funcionamiento esperado del programa.
2. Localice y descomprima el archivo esqueleto.zip.
3. Cree el proyecto en eclipse con el contenido del directorio n8_datacredito
4. Estudie la documentación del problema disponible en el directorio docs/specs del proyecto
   - La descripción del problema en Descripcion.doc.
   - Los requerimientos funcionales en RequerimientosFuncionales.doc
   - El modelo del mundo en modeloConceptual.jpg
   - El modelo de la interfaz en interfaz.jpg
   - El modelo de pruebas en pruebas.jpg
5. Asegúrese de tener activado el uso de aserciones para la ejecución del programa. Ver el tutorial en https://www.youtube.com/watch?v=Qt0cIQmflPw
6. Defina una configuración de ejecución del proyecto que verifique las invariantes de las clases del mundo. Para esto, debe indicar como parámetro a la JVM la opción –ea, tal como muestra la siguiente figura



7. Realice el mismo proceso en cada una de las configuraciones de Junit (AntecedenteTest, PersonaTest, TipoAntecedenteTest y SistemaFinancieroTest)

Proceso de desarrollo

Los siguientes pasos conforman el plan sugerido para desarrollar el ejercicio. La idea central es ir desarrollando y probando incrementalmente los métodos de las clases.

Ud. debe verificar que todos los métodos de todas las clases queden definidos correctamente.

Parte 1: Construcción y prueba de invariantes

1. Defina el invariante de la clase **Persona**, con base en la descripción del problema y el modelo conceptual propuesto.
2. Documente el invariante de la clase **Persona** en la cabecera de la clase, siguiendo las normas explicadas para ello.

3. Cree en la clase **Persona** el método **private void verificarInvariante()**, utilizando aserciones para validar el invariante definido en el punto anterior. Si lo considera necesario, implemente métodos privados para hacer verificaciones internas.
4. Defina el invariante de la clase **SistemaFinanciero** con base en la descripción del problema y el modelo conceptual propuesto
5. Documente el invariante de la clase **SistemaFinanciero** en la cabecera de la clase, siguiendo las normas explicadas para ello.
6. Cree en la clase **SistemaFinanciero** el método **private void verificarInvariante()** y utilice aserciones para validar el invariante que definió en el punto anterior. Si lo considera necesario, implemente métodos privados para hacer verificaciones internas.

A partir de este punto, usted debe utilizar los métodos para verificar los invariantes de las clases en todos aquellos métodos que modifiquen sus estados.

Parte 2: Creación de nuevas excepciones

1. Revise el modelo del mundo en docs\specs\ModeloConceptual.JPG e identifique las clases del mundo. Detecte las clases que representan las excepciones y sus relaciones con el resto del modelo.
2. Declare en java una clase por cada una de las excepciones faltantes que se manejan en el programa. Estas son: PersonaPreexistente, PersistenciaException, AntecedenteNoExisteException y FormatoInvalidoArchivoException.

Parte 3: Implementación de persistencia por serialización-deserialización

1. Guardar el estado actual (serialización):
   - Efectúe las modificaciones necesarias en las clases del mundo (SistemaFinanciero, Persona, Antecedente y TipoAntecedente) para que sus objetos puedan ser serializados.
   - Cree el método guardarSistemaFinanciero de la clase SistemaFinanciero encargado de salvar la información del SistemaFinanciero en un archivo binario. Este método debe lanzar una excepción de tipo PersistenciaException si hay problemas guardando los archivos.
   - Implemente el método dispose de la clase InterfazSistemaFinanciero para que se encargue de salvar la información del sistema financiero durante el proceso de cierre de la ventana principal. Si se presenta una excepción en el proceso de serialización del estado del modelo del mundo, este método debe avisar al usuario que la aplicación va a cerrarse sin salvar la información.
2. Construir el estado inicial (deserialización):
   - Complete el método cargarSistemaFinanciero de la clase SistemaFinanciero, que recibe como parámetro el nombre del archivo a partir del cual debe recuperar el estado del sistema finaciero. Este método debe considerar dos casos: si el archivo no existe, entonces el sistema financiero se crea vacío y su estado se guarda en el archivo indicado (puede utilizar el método guardarSistemaFinanciero para esto…). Si el archivo existe, entonces de ahí se extrae la información de las personas y sus antecedentes. Este método debe lanzar una excepción de tipo PersistenciaException si se encuentran problemas cargando los datos del archivo.
   - Complete el método importarDatos, que se va encargar de leer de un archivo plano de texto los antecedentes.

- En SistemaFinancieroTest, documente y construya todos los casos de prueba necesarios para verificar el correcto funcionamiento del constructor de SistemaFinanciero: Esto significa que cuando las cosas van bien el mundo queda cargado correctamente y cuando las cosas van mal, el constructor debe lanzar las excepciones correctas. Construya tantos casos de prueba como considere necesarios.

  Llame a estos métodos testSistemaFinanciero1(), testSistemaFinanciero2(), etc.
  Ayuda: Los métodos setupEscenario1 y setupEscenario2 tienen elementos interesantes que podrían ser útiles para el problema propuesto.

Parte 4: Uso de excepciones

1. Revise la documentación del método insertarPersona de la clase SistemaFinanciero e impleméntelo.
2. Implemente en la clase de pruebas SistemaFinancieroTest los métodos de prueba para verificar el correcto funcionamiento del método insertarPersona de la clase SistemaFinanciero implementado en el punto anterior. Estos son:
   - public void testInsertarPersona( ), el cual verifica el método que agrega una persona al SistemaFinanciero para el caso en el que no hay error. En este caso los datos de la persona que se agrega son correctos.
   - public void testInsertarPersonaPreexistente( ), el cual verifica el método que agrega una persona al SistemaFinanciero para el caso en el que una persona con dicha cédula ya existe. Este método debe probar que efectivamente el método insertarPersona arroja una excepción del tipo ExcepcionPersonaPreexistente.
3. Revise la documentación del método darAntecedente de la clase Persona e impleméntelo.
4. Revise la documentación del método eliminarAntecedente de la clase Persona e impleméntelo.
5. Verifique en este punto que los siguientes métodos de prueba se ejecutan sin errores:
   - testInsertarAntecedente de la clase de prueba PersonaTest
   - testSistemaFinanciero1 de la clase de prueba SistemaFinancieroTest
   - testSistemaFinanciero2 de la clase de prueba SistemaFinancieroTest

     ... Tantos como haya definido
   - testInsertarPersona de la clase de prueba SistemaFinancieroTest
   - testInsertarPersonaPreexistente de la clase de prueba SistemaFinancieroTest
   - testInsertarPersonaSistemaVacio de la clase de prueba SistemaFinancieroTest

Parte 5: Exportación de información (generación de reportes)

1. Revise la documentación del método generarReporte de la clase SistemaFinanciero e impleméntelo.

Parte 6: Creación de listas plegables en la interfaz

Complete la clase PanelPersonaAntecedente:

1. Cree los atributos listaPersonas (JList) y scrollListaPersonas (JScrollPane)
2. Cree el atributo comboAntecedentes de tipo JComboBox
3. Busque en el código del constructor el sitio donde deben ir las instrucciones que crean estos atributos y algunas instrucciones adicionales.

- Para que se tengan en cuenta los eventos del usuario, no olvide adicionar los listener apropiados: ActionListener para comboAntecedentes y ListSelectionListener para la lista de personas.

4. Complete el método public void valueChanged (ListSelectionEvent e) que cambia la información de antecedentes (el JComboBox) que se está mostrando a los antecedentes de la persona que se acaba de seleccionar. Ayuda: Hay métodos de la clase InterfazSistemaFinanciero, muy útiles para esto.

Parte 7: Prueba general de la aplicación

Verifique el correcto funcionamiento de toda la aplicación, realizando todas las operaciones provistas por el sistema. En particular, revise que los reportes generados corresponden a la información que Uds. han ingresado al sistema, mirando el archivo de texto generado y comparándolo con la información que tienen en pantalla. Cómo sería el método que hace la prueba automática de este método?