Упражнение 6-1. Обработка исключительных ситуаций.

Цель упражнения: Изучить способы обработки исключительных ситуаций в программах на `Java`.

Описание упражнения: В этом упражнении вы объявите в проекте несколько собственных типов исключений и используете их в методах классов проекта.

1) Объявите в пакете `ru.itmo.exceptions` следующие классы исключений:
   a) `CatalogLoadException` : исключение, выбрасываемое методом `load` интерфейса `CatalogLoader`. Выбрасывается при любых ошибках загрузки каталога.
   b) `ItemAlreadyExistsException` : исключение, выбрасываемое методом `addItem` класса `ItemCatalog`. Выбрасывается в том случае, когда добавляемая позиция уже присутствует в каталоге.
   c) `NegativeQuantityException` : исключение, выбрасываемое методом `remQuantity` класса `Warehouse` (будет создан в последующих упражнениях). Выбрасывается в случае попытки списать со склада количество товара больше имеющегося.

      Примечание: Все эти классы являются проверяемыми исключениями и должны наследовать классу `Exception`.
2) Предусмотрите в методе `load` интерфейса `CatalogLoader` и реализующего его класса `CatalogStubLoader` выброс исключения `CatalogLoadException`. Для класса `CatalogStubLoader` код должен предусматривать ошибки на этапе добавления позиции в каталог:

```
try {
   cat.addItem(item1);
   cat.addItem(item2); }
catch (ItemAlreadyExistsException e) { // TODO Auto-generated catch
block
   e.printStackTrace();
   throw new CatalogLoadException(e);
}
```

3) Предусмотрите в методе `addItem` класса `ItemCatalog` выброс исключения `ItemAlreadyExistsException`. Исключение должно выбрасываться в том случае, когда добавляемая позиция уже есть в каталоге.
4) Исправьте все ошибки, появившиеся в проекте в связи с необходимостью предусмотреть обработку новых исключений.
5) Запустите проект и проверьте его работоспособность. Протестируйте ситуации, в которых могут быть выброшены, добавленные в проект исключения.

Упражнение 6-2 Синхронизация потоков.

Цель упражнения: Изучить способы синхронизации работы нескольких потоков в многопоточной среде JVM.

Описание упражнения: В этом упражнении вы смоделируете работу банковских операций со счетами.

1) Добавьте в проект новый пакет `sync`, в котором буде создавать все классы из этого упражнения
2) Создайте класс `U1901Bank`, в котором будем имитировать работу банковской операции со счетами и наполните его следующим кодом:
   a) Объявите две `int`-переменные уровня экземпляра класса с именами `intTo` и `intFrom`, которые будут имитировать счет-отправитель и счет-получатель (кредит и дебет). Переменную `intFrom` инициализируйте значением 220.
   b) Создайте метод без возвращаемого значения с именем `calc`. В этом методе будет организована банковская операция по переброске денег между счетами (между переменными `intFrom` и `intTo`). Между снятием денег с одного счета (уменьшением значения переменной `intFrom`) и пополнением другого счета (увеличением переменной `intTo`) будет организована задержка. длительность которой передается в качестве входного параметра. Также в качестве входного параметра передается сумма перевода между счетами. Наполните метод следующим кодом:
      i) Укажите два входных параметра – `intTransaction` (типа int) для передачи суммы и `lngTimeout` (типа `long`) для указания длительности временной задержки.
      ii) Для контроля работы метода выведите сообщение, в котором укажите начальные значения переменных `intTo` и `intFrom`, а также имя текущего потока (при помощи метода `Thread.currentThread().getName()`.
      iii) Уменьшите значение переменной `intFrom` на значение переменной `intTransaction` и сохраните результат в той же переменной `intFrom`
      iv) Организуйте временную задержку на `lngTimeout` миллисекунд при помощи метода `Sleep` класса `Thread`. Оберните вызов этого метода в `try/catch`.
      v) Увеличьте значение переменной `intTo` на значение переменной `intTransaction` и сохраните результат в той же переменной `intTo`
      vi) В завершении выведите имя текущего потока и изменившиеся значения переменных (подобно пункту 2.ii). Рекомендуется в обоих случаях указать какое-нибудь уникальное слово (например `before`/`after`) для идентификации вывода.
3) Создайте класс-наследник от `Thread` с именем `U1901Thread`, в котором будем реализовывать многопоточность для работы с методом `calc` класса `U1901Bank`. Наполните класс следующим кодом:
   a) Объявите следующие переменные уровня экземпляра класса:
      i) `bankWork` типа `U1901Bank`

ii) `intTrans` типа `int`

iii) `lngSleep` типа `long`.

b) Создайте конструктор, в котором должно быть три параметра для заполнения этих переменных уровня экземпляра класса. Тип параметров должен соответствовать типу переменной, имена параметров можете указать сами.

Такой извилистый способ требуется потому, что метод `run()`, который используется для многопоточной работы, не имеет параметров.

c) Создайте метод `run()`, в коде которого вызовите метод `calc` объекта `bankWork` и передайте в этот метод значения переменных `intTrans` и `lngSleep`.

4) Создайте класс `U1901Main`, который будет вызываться из-под `JVM` для проверки работоспособности многопоточности. В этом классе будет один метод, имя его предлагается определить самостоятельно. В этом методе делаем следующее:

a) Создать экземпляр класса `U1901Bank` с именем `bankMain` и инициализировать его конструктором.

b) Создайте экземпляр класса `U1901Thread` с именем `threadOne` и инициализируйте его конструктором со следующими параметрами: `bankMain, 100,2000`.

c) Задайте этому экземпляру уникальное и понятное имя потока при помощи метода `setName` и приоритет `Thread.MAX_PRIORITY` при помощи метода `setPriority`.

d) Запустите поток при помощи метода `start()`

e) Повторите действие пунктов `b,c,d` для экземпляра того же класса, но с именем threadTwo, другим именем потока, конструктором с параметрами: bankMain, 50,300.

f) В качестве контрольного выстрела, в конце метода можно вывести имя текущего потока – `Thread.currentThread().getName()`. Имена всех трех потоков должны быть разными!

5) Запустите класс U1901Main. Вывод должен быть примерно таким -

```
before Thread=thread_100, From=220, To=0
before Thread=thread_50, From=220, To=0
End, main
After Thread=thread_50, From=70, To=50
After Thread=thread_100, From=70, To=150
```

Что здесь не правильно – в выделенной строке сумма значений переменных отличается от суммы в других строках. Это плохо – деньги где-то зависли. Хозяева могут дать по шее.

6) Для исправления сего недостатка надо сделать метод `calc` синхронизированным. После этого вывод будет таким:

```
End, main
before Thread=thread_100, From=220, To=0
After Thread=thread_100, From=120, To=100
before Thread=thread_50, From=120, To=100
After Thread=thread_50, From=70, To=150
```

7) Ура, мы победили – у нас везде `220`!

Упражнение 6-3-1 Потоки ввода -вывода.

Цель упражнения: Научиться работать с потоками ввода-вывода.

Описание упражнения: В этом упражнении вы реализуете загрузчик каталога товаров из структурированного текстового файла.

1) Добавьте в пакет `ru.itmo.client` новый класс `CatalogFileLoader`, реализующий интерфейс `CatalogLoader`.
2) Добавьте в класс `CatalogFileLoader` строковое поле, содержащее имя файла со списком товаров и конструктор, принимающий на вход имя этого файла:

```
private String fileName;
public CatalogFileLoader(String fileName) {
   this.fileName = fileName;
}
```

3) Реализуйте метод `load` класса `CatalogFileLoader` следующим образом:

```
File f = new File(fileName);
FileInputStream fis;
String line;
 try {
   fis = new FileInputStream(f);
   Scanner s = new Scanner(fis);

   while(s.hasNextLine()){
         line = s.nextLine();
         if(line.length()==0) break;
         String[] item_fld = line.split(";");
         String name = item_fld[0];
         float price = Float.parseFloat(item_fld[1]);
         short expires = Short.parseShort(item_fld[2]);
         FoodItem item = new FoodItem(name,price,null, new
Date(),expires);
         cat.addItem(item);
   }
 } catch (FileNotFoundException e) {
   e.printStackTrace();
   throw new CatalogLoadException(e);
 } catch (ItemAlreadyExistsException e) {
   e.printStackTrace();
   throw new CatalogLoadException(e);
```

```
  }
}
```

4) Используйте, предоставленную инструктором, версию файла `items.lst` для загрузки каталога товаров.

Упражнение 6-3-2 (Опционально). Потоки ввода -вывода.

1) Если еще остались силы и желание, выполните чтение данных из текстовых файлов с кодировками UTF-8 и WINDOWS-1251. Байты, считанные из файлов преобразовать в символы и вывести на экран.