# Упражнение 4-1.Классы-коллекции.

**Цель упражнения:** Изучить преимущества использования классов-коллекций, предоставляемых в стандартной поставке JDK.

**Описание упражнения:** В этом упражнении вы реализуете класс ItemCatalog, хранящий список товаров, продаваемых интернет-магазином. Список в дальнейшем будет использоваться для демонстрации товаров покупателям на сайте магазина.

1) Добавьте в проект класс ItemCatalog, в котором будет реализована логика хранения списка товаров.
2) Включите в класс ItemCatalog поля, определенные следующим образом:

```
private HashMap<Integer,GenericItem> catalog =
                                new HashMap<Integer,GenericItem>();
private ArrayList<GenericItem> ALCatalog =
                                new ArrayList<GenericItem>();
```

*Примечание:* Обе эти коллекции будут хранить один и тот же список товаров. Коллекция HashMap более оптимальна для последующего поиска товаров в каталоге, а коллекция ArrayList нужна для сравнения с ней.

3) Реализуйте в классе ItemCatalog следующие методы:
   a) `public void addItem(GenericItem item)` : добавляет товар в каталог

```
public void addItem(GenericItem item) {
      catalog.put(item.ID, item); // Добавляем товар в HashMap
      ALCatalog.add(item); // Добавляем тот же товар в ArrayList
}
```

   b) `public void printItems()` : распечатывает товары из каталога на экране. Распечатку следует производить с использованием метода toString класса GenericItem.

```
 public void printItems(){
 for(GenericItem i : ALCatalog){
      System.out.println(i);
  }
}
```

   c) `public GenericItem findItemByID(int id)` : производит поиск в каталоге по переданному id товара. Поиск следует производит в коллекции catalog типа HashMap

```
public GenericItem findItemByID(int id){
//Если нет такого ID, возвращаем пустое значение
      if(!catalog.containsKey(id)) {
            return null;
      } else{
```

```
            return catalog.get(id);
        }
    }
```

d) `public GenericItem findItemByIDAL(int id)` : производит поиск в каталоге по переданному `id` товара. Поиск следует производит в коллекции `ALCatalog` типа `ArrayList`

```
public GenericItem findItemByIDAL(int id){
        for(GenericItem i : ALCatalog){
            if(i.ID==id) return i;
        }
        return null;
    }
```

4) В методе `main` класса `Main` создайте новый экземпляр класса `ItemCatalog`. С помощью метода `addItem` добавьте в него несколько (порядка `10`) товаров.

5) С помощью приема из упр. 3-1 сравните скорость поиска по двум типам коллекций:

```
long begin = new Date().getTime();

for(int i=0; i<100000;i++)
cat.findItemByID(10);
long end = new Date().getTime();
System.out.println("In HashMap: "+(end-begin)); begin = new
Date().getTime();
for(int i=0; i<100000;i++)
   cat.findItemByIDAL(10);
end = new Date().getTime();
System.out.println("In ArrayList: "+(end-begin));
```

*Примечание:* Поскольку у нас нет возможности сформировать действительно большие списки товаров, приходится производить большое количество циклов поиска, чтобы разница во времени накапливалась и становилась ощутимой.

# Упражнение4-2. Абстрактные классы и интерфейсы.

**Цель упражнения:** Изучить полезные свойства абстрактных классов и интерфейсов.

**Описание упражнения:** В этом упражнении вы добавите в проект интерфейс `CatalogLoader` с методом `load`. Объект, реализующий метод `load` будет способен загружать список товаров в указанный каталог (`ItemCatalog`). В дальнейшем в нашем проекте появятся несколько классов реализующих метод `load` и заполняющих каталог из различных источников.

1) Добавьте в проект новый интерфейс `CatalogLoader`.

2) Опишите в интерфейсе `CatalogLoader` следующий метод:

```
public void load(ItemCatalog cat);
```

3) Добавьте в проект класс `CatalogStubLoader`, реализующий интерфейс `CatalogLoader`.

4) В методе `load` класса `CatalogStubLoader` реализуйте «ручной» способ загрузки каталога с помощью явно созданных программистом объектов:

```
GenericItem item1 = new GenericItem("Sony TV",23000,Category.GENERAL);
FoodItem item2 = new FoodItem("Bread",12,null,new Date(),(short)10);
cat.addItem(item1);
cat.addItem(item2);
```

5) В методе `main` класса `Main` создайте новый фрагмент кода, загружающий товары в каталог с помощью объекта-загрузчика:

```
CatalogLoader loader = new CatalogStubLoader();
loader.load(cat);
```

6) Запустите программу и проверьте корректность загрузки списка товаров.

## Упражнение 4-3. Пакеты, модификаторы доступа и инкапсуляция.

**Цель упражнения:** Изучить на практике использование механизма пакетов и принципа инкапсуляции

**Описание упражнения:** В этом упражнении вы создадите пакеты в рамках проекта `StockListProject` и распределите классы и интерфейсы проекта по пакетам. Также вы создадите инкапсулированные версии классов проекта.

1) Инкапсулируйте классы `GenericItem`, `FoodItem`, `TechnicalItem` и `ItemCatalog` своего проекта. Для этого объявите все их поля как `private`, а для доступа к ним создайте соответствующие методы `set…` и `get…`
   Это лучше сделать при помощи `eclipse` – `Source`/`Generate Setters and Getters`.

2) Создайте в проекте `StockListProject` пакеты со следующим содержанием:
   a) `ru.billing.client`
      i) Класс `Main`
      ii) Интерфейс `CatalogLoader`
      iii) Класс `CatalogStubLoader`
   b) `ru.billing.exceptions`
   c) `ru.billing.stocklist`
      i) Перечисление `Category`
      ii) Класс `GenericItem`
      iii) Класс `FoodItem`

iv) Класс `TechnicalItem`

v) Класс `ItemCatalog`

d) `ru.lanit.warehouse`

3) Исправьте ошибки, появившиеся в проекте из-за необходимости импорта классов из других пакетов.

4) Запустите программу, проверьте работоспособность проекта.