Vending Machine

Your task is to create a VendingMachine (candy vending machine) in Java. It should be able to receive money in predetermined amounts. Of course, you should also be able to decide which product to buy by choosing the product from a list of choices. The machine should also be able to give back change.

Functionality requirements:

- The following denominations must be accepted by the vending machine:
  - 1kr, 2kr, 5kr, 10kr, 20kr, 50kr, 100kr, 200kr, 500kr, 1000kr.
- Money that is put in and accepted by the vending machine must be added to a deposit pool. The deposit pool should represent means of payment for the product.
- The user should be able to buy several products from the vending machine if there is enough money in the deposit pool.
- When the user decides to stop buying things, what is left of the deposit pool should be returned to the user as a change.
- The machine must have at least three types of products in it. (drinks, food, sweets, fruits etc.)

Code requirements:

- Accepted denominations should be represented as an integer array or as enum types.
- Each type of product should **inherit** from the same **abstract class** (Product) **or interface**. This superclass should define the following common functionality:
  - **String examine()**, show product info. For example, price, name, calories, allergens, etc. (Return String)
  - **String use()**, use / consume the product. (Return String)
- An **interface** VendingMachine should be created that defines the following methods:
  - **void addCurrency (int amount)** - Add to the deposit pool (moneyPool).
  - **Product request (int productNumber)** - Buy a Product.
  - **int endSession ()** Returns change and resets the deposit pool.
  - **String getDescription (int productNumber)** - View a product description.
  - **int getBalance ()** - Returns the deposit pool amount.
  - **String [] getProducts ()** - Returns all Products' names and product numbers.
- A class that implements the VendingMachine interface.
  - Contains array with Products and deposit pool.

For approved:

- All functionality implemented.
- The classes tested with JUnit. 70% code coverage.

Optional:

- Build a user interface for the application (graphic or console) where the user will be able to communicate with all parts of the application.