First and foremost, thank you for your consideration and giving me this opportunity!

1. How to build and run this application
   a) Download from https://github.com/bigzj/VendingMachine or unzip the zip file to any folder
   b) Use Eclipse or any IDE to "import existing Maven project", or build from command line using Maven
   c) Run Unit Test from org.icann.vendingmachine.test.VendingMachineTest

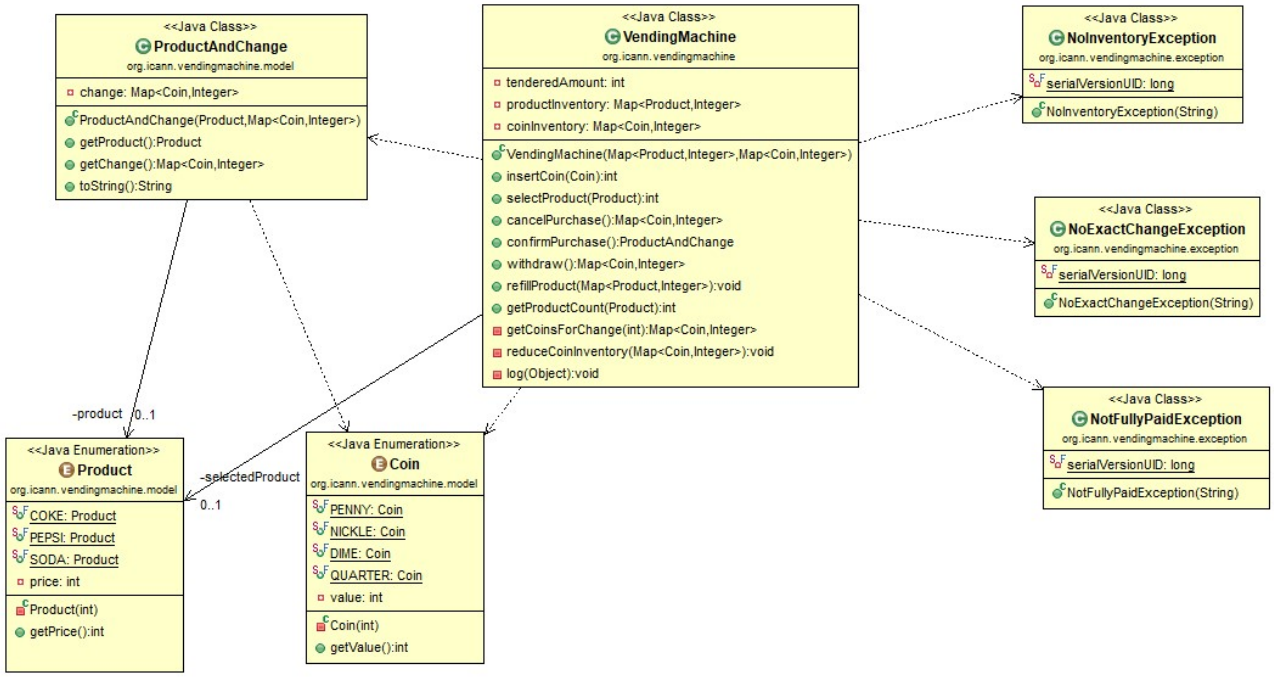
2. High Level Design
   a) Simplicity is given priority over using of various patterns, hierarchies, etc
   b) Core Java libraries are the only dependencies of the main application; whereas JUnit4 is used for Unit Testing.
   c) com.icann.vendingmachine – the main package of the application
      i. VendingMachine – the main class of the application
   d) com.icann.vendingmachine.model – the data models
      i. Product – Enumeration, contains all products that the machine sells.
      ii. Coin – Enumeration, contains all the coin types that the machine takes
      iii. ProductAndChange, encapsulates the return of the confirmPurchase() method, contains the product that the user selected and changes of the coins if any
   e) com.icann.vendingmachine.exception – the business exceptions
      i. NoInventoryException – thrown from the selectProduct() method if the product selected has no inventory
      ii. NoExactChangeException – thrown from the completePurchase() method if no exact change can be returned from the current coins that the machine has
      iii. NotFullyPaidException – thrown from the completePurchase() method if the tenderedAmount is less than the price of the selected product

3. Low Level Design
   a) Initialization – products and coins should be provided when creating the vending machine. Null values are accepted though, in that case, a vending machine with no product and/or coin will be created
   b) Enumeration is used for product and coin, so we do not have do validation of them all over the place
   c) Class diagram is on the next page

4. Assumptions
   a) No database is required; there is no mention of persistence in the requirement. To simplify the build / test process, I decided to not using a database. But a persistence layer can be easily added later
   b) It is a single threaded application, only one user will use the machine at a time, same as real world case
   c) If the previous user did not complete the purchase and left, e.g. inserted some coins and then left, the current user will be able to use the previously inserted coins; same as real world case

5. Unit Testing
   a) JUnit4 is used to perform the unit testing and it is the only dependency of the application besides core Java libraries
   b) All the public methods in the VendingMachine class are tested, including alternative flows (exceptional cases)
   c) The sequence of the 9 tests are important, so FixMethodOrder by the method name is used

Once again, thank you for your time and consideration!


<<Java Class>>
ProductAndChange
org.icann.vendingmachine.model
change: Map<Coin,Integer>
ProductAndChange(Product,Map<Coin,Integer>)
getProduct():Product
getChange():Map<Coin,Integer>
toString():String
<<Java Class>>
VendingMachine
org.icann.vendingmachine
tenderedAmount: int
productInventory: Map<Product,Integer>
coinInventory: Map<Coin,Integer>
VendingMachine(Map<Product,Integer>,Map<Coin,Integer>)
insertCoin(Coin):int
selectProduct(Product):int
cancelPurchase():Map<Coin,Integer>
confirmPurchase():ProductAndChange
withdraw():Map<Coin,Integer>
refillProduct(Map<Product,Integer>):void
getProductCount(Product):int
getCoinsForChange(int):Map<Coin,Integer>
reduceCoinInventory(Map<Coin,Integer>):void
log(Object):void
<<Java Class>>
NoInventoryException
org.icann.vendingmachine.exception
serialVersionUID: long
NoInventoryException(String)
<<Java Class>>
NoExactChangeException
org.icann.vendingmachine.exception
serialVersionUID: long
NoExactChangeException(String)
<<Java Class>>
NotFullyPaidException
org.icann.vendingmachine.exception
serialVersionUID: long
NotFullyPaidException(String)
-product
0..1
-selectedProduct
0..1
<<Java Enumeration>>
Product
org.icann.vendingmachine.model
COKE: Product
PEPSI: Product
SODA: Product
price: int
Product(int)
getPrice():int
<<Java Enumeration>>
Coin
org.icann.vendingmachine.model
PENNY: Coin
NICKLE: Coin
DIME: Coin
QUARTER: Coin
value: int
Coin(int)
getValue():int