Requirements

- A working vending machine that accepts payment from a card, dispenses and item and deducts the item's cost from the card's balance.
- A terminal that accepts cash from the user and converts it to points on the card at the ratio of 2 credits per $1.
- The terminal reads the card, and then the user can: check the card balance (along with their card #), print out the balance, and transfer credits to a different card.
- -At the time of the transaction, the terminal will read the card, ask for a selection, pull the selected food from the machine, and dispense it.

-If the balance is insufficient, the transaction will be declined, the user will be notified, and no credits will be deducted.

- If the balance is sufficient, the item will be dispensed
- At the end of the transaction, print: card#, item purchased, and the balance.

- A card offers limited storage
- A card has a specific card#
- A card has a balance of credits

Designing a Plan

- Card
  - Card #
    Current Credit balance
  - Balance can never be negative

- Terminal
  Accepts $ and converts to credits (2pts/$1)
  Money is accepted as a whole number.
  Can print out card balance + card #
  Transfer balance between 2 cards
  Display current VendingInventory
  Grabs the selected item from the inventory
  If the card balance is 0, do not dispense item, notify the user, and do not deduct any credits
  Prints out card#, card balance, # of remaining inventory of purchased item

- TerminalCalculations
  Adds, subtracts balance to/from the card based on the type of transaction

- VendingInventory
  Intakes a selection from the terminal
  Stores a collection of food items
  Each food item will have a set price
  The vending machine will keep track of the # of food items it has in stock
  If the stock for a specific item goes to 0, notify the user (this should probably happen in the terminal)

Testing	**Implementing the Plan**

Instantiate 2 cards and whatever other objects might be necessary to test your program.
• Load credits onto each card
  • Card for user 1 loaded with 100 credits
  • Card for user 2 loaded with 120 credits
• Purchase a bunch of items using both cards.
  • Card for user 1 purchased several items (Chips, Cookies, Water, Soda, and Granola Bar)
• Transfer the balance of credits from Card 1 to Card 2.
  • 60 credits from card for user 1 transferred to Card 2, making new balance for card 2 - 140.
  • 60 credits from card for user 2 transferred to card 1, making new balance for card 1 - 100
•
	-All fields must be private.
	
	Card:
	private static Integer cardNumberIndex = 1;
	private Integer cardNumber;
	private Integer balance;
	Create parameterized constructor for Card (Integer Balance)
	Create constructors for getCardNumber, getBalance and setBalance
	
	Inventory:
	Properties:
	VendingItemsEnum name, Integer price, Integer stockChips, Integer stockCookies, Integer stockGranolaBars, Integer stockWater, Integer stockSoda
	Create parameterized constructor for Inventory (VendingItemsEnum name, Integer price)
	Create constructors for getName, getPrice, loadStock, getStockChips, getStockCookies, getStockWater, getStockSoda, getStockGranolaBars, purchaseChips, purchaseCookies, purchaseGranolaBars, purchaseWater, and purchaseSoda
	
	Vending Items ENUM:
	CHIPS, COOKIES, WATER, SODA, GRANOLA_BARS

	Main (Terminal) Scanner for user input Main (Class) Create instance of using the Card class cardForUser1 and cardForUser2 Methods created for the following addCredits transferCredits checkBalance makePurchase (utilize switch statements for the 5 items loaded into the vending machine) Start (starting point for the vending machine using switch statements for each method created to operate the vending machine)