You are a user. You are in the market to purchase, sell, etc. your stocks. In the market, there is the list of stocks available that you can buy or sell.

There are 2 stock sets available for stocks

1. Original Listing of Stocks
2. Portfolio Listing of Stocks

Both of these lists have 2 attributes

- capacity: the max no. of stocks possible
- actual stocks: the total no. of available stocks

For Stock
In the original list, you have to maintain the following info

- Name (String)
- Price (double)

If a user buys a stock, then it becomes part of that user's portfolio. Apart from Name and Price, you have to keep info of

- Quantity (int)
- TotalStockPrice = Price*Quantity (double)

Functionalities to implement

- **Add Stock: Adds a new stock to the original list**
    - Input parameters: String name, double price
    - If the addition exceeds capacity then print the message, "Stock can't be purchased".
    - Output format:
        - exception message (if any)
        - call Display Stocks
- **Add Stock in Portfolio: Adds a new stock to the portfolio list**
    - Input parameters: String name, int quantity
    - You should only be able to add if that stock exists in the original list.
    - If it's an invalid stock then print the message, "Invalid stock. Add operation can't be performed!"
    - If the addition exceeds capacity then print the message, "Stock can't be purchased".
    - Output format:
        - exception message (if any)
        - call Display Portfolio Stocks
- **Display Stocks**
    - Print the name and price of all the stocks in the original list.

    - Please follow the below format:
      Stock Name: Apple
      Stock Price: 100
- **Display Portfolio Stocks**
    - Print the name and price of all the stocks in the portfolio list.
    - Please follow the below format:
      Stock Name: Apple
      Stock Price: 100
      Stock Quantity: 5
      Stock Total Price: 500

- **Purchase Stock**
    - Input parameters: String name, int quantity
    - Check whether the stock user wants to purchase is valid i.e exists in the original list.
        - if invalid then print "Invalid stock. Purchase operation can't be performed!"
    - Purchase of stock can be done for new as well as existing stocks.
    - If it exceeds capacity, the print "Stock can't be purchased".
    - Output format:
        - exception message (if any)
        - call Display Portfolio Stocks
- **Sell Stock**
    - Input parameters: String name, int quantity
    - First check- whether the stock user wants to sell is valid i.e., exists in the original list.
        - if invalid then print "Invalid stock. Sell operation can't be performed!"
    - Second check- whether the stock exists in the portfolio.
        - If doesn't exist then print "Stock doesn't exist".
    - If both checks are true, then only selling can be done.
    - If selling is not possible because of less quantity, the print "Invalid stock quantity.".
    - Output format:
        - exception message (if any)
        - call Display Portfolio Stocks

Keep the following sequence of operations in the switch case

1. Add stock to the original stocks list
2. Add stock to the portfolio
3. Display the original stocks list
4. Display portfolio stocks list
5. Purchase portfolio stocks

6. Sell portfolio stocks

Note: Please make sure you are taking all inputs in your main method, or else you will get compilation errors.

Sample Test Case 1
Input
```
4         -- capacity of original list
3         -- actual size of the original list
apple   -- stock 1's name
100       -- stock 1's price
google   -- stock 2's name
200       -- stock 2's price
tesla     -- stock 3's name
50        -- stock 3's price
3         -- capacity of the user's portfolio
2         -- actual size of the user's portfolio
apple     -- stock 1's name
2         -- stock 1's quantity
google   -- stock 2's name
1         -- stock 2's quantity
3         -- option 3: display all stocks of original list
```

Output
```
Stock Name: apple
Stock Price: 100.0
Stock Name: google
Stock Price: 200.0
Stock Name: tesla
Stock Price: 50.0
```

Explanation
All stock of the original list is displayed in the desired format.

Sample Test Case 2
Input
```
4
4
apple
100
google
200
tesla
50
```

```
amazon
100
3
2
apple
2
google
1
1          -- option 1: add stock to the original list
ola         -- new stock name
250       -- new stock price
```

Output

```
Stock can't be purchased
Stock Name: apple
Stock Price: 100.0
Stock Name: google
Stock Price: 200.0
Stock Name: tesla
Stock Price: 50.0
Stock Name: amazon
Stock Price: 100.0
```

Explanation

Message (Stock can't be purchased) printed as original list stock capacity is equal to its size. All the stocks of the original list are displayed.

Solution

```
import java.util.Scanner;

class Stock {
  String name;
  double price;
  int quantity;
  double totalPrice;

  public Stock(String name, double price) {
    this.name = name;
    this.price = price;
  }

  public Stock(String name, double price, int quantity, double totalPrice) {
    this.name = name;
    this.price = price;
    this.quantity = quantity;
    this.totalPrice = totalPrice;
  }
}

class OriginalStock {
  int capacity;
  int size;
  Stock[] stocks;

  public OriginalStock(int orgCapacity, int orgSize) {
    capacity = orgCapacity;
    size = orgSize;
    stocks = new Stock[capacity];
  }

  public void addStock(String name, double price) {
    if (capacity == size) {
      System.out.println("Stock can't be purchased");
      return;
    }
    stocks[size] = new Stock(name, price);
    size++;
  }

  public void display() {
    for (int i = 0; i < size; i++) {
      System.out.println("Stock Name: " + stocks[i].name);
      System.out.println("Stock Price: " + stocks[i].price);
    }
  }

  public Stock getStock(String name) {
    for (int i = 0; i < size; i++)
      if (stocks[i].name.equals(name))
        return stocks[i];
    return null;
  }

}

class PortfolioStock {
  int capacity;
  int size;
  Stock[] stocks;

  OriginalStock originalStock;

  public PortfolioStock(int portCapacity, int portSize, OriginalStock
originalStock) {
    capacity = portCapacity;
    size = portSize;
    stocks = new Stock[capacity];
    this.originalStock = originalStock;

  }

  public void addStock(String name, int quantity) {
    Stock isExist = originalStock.getStock(name);
    if (isExist == null)
      System.out.println("Invalid stock. Add operation can't be performed!");
    else if (capacity == size)
      System.out.println("Stock can't be purchased");
    else {
      stocks[size] = new Stock(name, isExist.price, quantity, isExist.price *
quantity);
      size++;
    }
  }

  public void display() {
    for (int i = 0; i < size; i++) {
      System.out.println("Stock Name: " + stocks[i].name);
      System.out.println("Stock Price: " + stocks[i].price);
      System.out.println("Stock Quantity: " + stocks[i].quantity);
      System.out.println("Stock TotalPrice: " + stocks[i].totalPrice);
    }
  }

  public void purchase(String name, int quantity) {
    Stock tmp = originalStock.getStock(name);
    if (tmp == null)
      System.out.println("Invalid stock. Purchase operation can't be
performed!");
    else if (capacity == size)
      System.out.println("Stock can't be purchased");
    else {
      Stock curr = isExistInPortfolio(name);
      if (curr != null) {
        curr.quantity += quantity;
        curr.totalPrice = curr.quantity * curr.price;
      } else {
        stocks[size] = new Stock(name, tmp.price, quantity, tmp.price *
quantity);
        size++;
      }
    }

  }
```

```java
  private Stock isExistInPortfolio(String name) {
    for (int i = 0; i < size; i++)
      if (stocks[i].name.equals(name))
        return stocks[i];
    return null;
  }

  public void sell(String name, int quantity) {
    Stock tmp = originalStock.getStock(name);
    if (tmp == null) {
      System.out.println("Invalid stock. Sell operation can't be performed!");
      return;
    }
    Stock curr = isExistInPortfolio(name);
    if (curr == null)
      System.out.println("Stock doesn't exist");
    else {
      if (curr.quantity < quantity)
        System.out.println("Invalid stock quantity");
      else {
        curr.quantity -= quantity;
        curr.totalPrice = curr.price * curr.quantity;
      }
    }

  }
}

public class Main {
  public static void main(String[] args) {
    Scanner in = new Scanner((System.in));
    int orgCapacity = in.nextInt();
    int orgSize = in.nextInt();
    OriginalStock originalStock = new OriginalStock(orgCapacity, orgSize);
    for (int i = 0; i < orgSize; i++) {
      String name = in.next();
      double price = in.nextDouble();
      Stock tmp = new Stock(name, price);
      originalStock.stocks[i] = tmp;
    }
    int portCapacity = in.nextInt();
    int portSize = in.nextInt();
    PortfolioStock portfolioStock = new PortfolioStock(portCapacity, portSize,
originalStock);
    for (int i = 0; i < portSize; i++) {
      String name = in.next();
      Stock curr = originalStock.getStock(name);
      double price = curr.price;
      int quantity = in.nextInt();
      double totalPrice = price * quantity;
      Stock tmp = new Stock(name, price, quantity, totalPrice);
      portfolioStock.stocks[i] = tmp;
    }
    int choice = in.nextInt();
    switch (choice) {
    case 1:
      String name = in.next();
      originalStock.addStock(name, in.nextDouble());
      originalStock.display();
      break;

    case 2:
      portfolioStock.addStock(in.next(), in.nextInt());
      portfolioStock.display();
      break;

    case 3:
      originalStock.display();
      break;

    case 4:
      portfolioStock.display();
      break;

    case 5:
      portfolioStock.purchase(in.next(), in.nextInt());
      portfolioStock.display();
      break;

    case 6:
      portfolioStock.sell(in.next(), in.nextInt());
      portfolioStock.display();
      break;

    }

  }
}
```