```
# Python Fashion Shop Assignment

import pickle

from BTCInput import *

class StockItem(object):

    show_instrumentation = False

    min_price = 0.5
    max_price = 500.0

    max_stock_add = 50

    def __init__(self, stock_ref, price, color, location):
        if StockItem.show_instrumentation:
            print('** StockItem __init__ called')
        self.stock_ref = stock_ref
        self.__price = price
        self.__stock_level = 0
        self.StockItem_version = 1
        self.color = color
        self.location = location

    @property
    def item_name(self):
        if StockItem.show_instrumentation:
            print('** StockItem get item_name called')
        return 'Stock Item'

    def check_version(self):
        if StockItem.show_instrumentation:
            print('** StockItem check_version called')

        pass

    def __str__(self):
        if StockItem.show_instrumentation:
            print('** StockItem __str__ called')
        template = '''Stock Reference: {0}
Type: {1}
Location: {2}
Price: {3}
```

```
Stock level: {4}
Color: {5}'''

        return template.format(self.stock_ref, self.item_name, self.location,
                               self.price, self.stock_level, self.color)

    def add_stock(self,count):

        if StockItem.show_instrumentation:
            print('** StockItem add_stock called')
        if count < 0 or count > StockItem.max_stock_add:
            raise Exception('Invalid add amount')

        self.__stock_level = self.__stock_level + count

    def sell_stock(self, count):

        if StockItem.show_instrumentation:
            print('** StockItem sell_stock called')
        if count < 1:
            raise Exception('Invalid number of items to sell')

        if count > self.stock_level:
            raise Exception('Not enough stock to sell')

        self.__stock_level = self.__stock_level - count

    def set_price(self, new_price):
        if StockItem.show_instrumentation:
            print('** StockItem set_price called')
        if price < StockItem.min_price or price > StockItem.max_price:
            raise Exception('Price out of range')
        self.__price = new_price

    @property
    def price(self):
        if StockItem.show_instrumentation:
            print('** StockItem get price called')
        return self.__price

    @property
    def stock_level(self):
        if StockItem.show_instrumentation:
            print('** StockItem get stock level called')
```

```
        return self.__stock_level


class Dress(StockItem):

    def __init__(self, stock_ref, price, color, pattern, size, location):
        if StockItem.show_instrumentation:
            print('** Dress __init__ called')
        super().__init__(stock_ref=stock_ref, price=price,
                         color=color, location=location)
        self.pattern = pattern
        self.size = size
        self.Dress_version = 1

    @property
    def item_name(self):
        if StockItem.show_instrumentation:
            print('** Dress get item_name called')
        return 'Dress'

    def check_version(self):
        if StockItem.show_instrumentation:
            print('** Dress check_version called')

        super().check_version()
        pass

    def __str__(self):
        if StockItem.show_instrumentation:
            print('** Dress __str__ called')
        stock_details = super().__str__()
        template = '''{0}
Pattern: {1}
Size: {2}'''
        return template.format(stock_details, self.pattern,
                               self.size)


class Pants(StockItem):

    def __init__(self, stock_ref, price, color, pattern, length, waist, location):
        if StockItem.show_instrumentation:
            print('** Pants __init__ called')
        super().__init__(stock_ref=stock_ref, price=price,
```

```
                color=color, location=location)
        self.pattern = pattern
        self.length = length
        self.waist = waist
        self.pants_version = 1

    @property
    def item_name(self):
        if StockItem.show_instrumentation:
            print('** Pants get item_name called')
        return 'Pants'

    def check_version(self):
        if StockItem.show_instrumentation:
            print('** Pants check_version called')

        super().check_version()
        pass

    def __str__(self):
        if StockItem.show_instrumentation:
            print('** Pants __str__ called')
        stock_details = super().__str__()
        template = '''{0}
Pattern: {1}
Length: {2}
Waist: {3}'''
        return template.format(stock_details, self.pattern,
                               self.length, self.waist)


class Jeans(Pants):

    def __init__(self, stock_ref, price, color, pattern, length, waist, style, location):
        if StockItem.show_instrumentation:
            print('** Jeans __init__ called')
        super().__init__(stock_ref=stock_ref, price=price,
                         color=color, pattern=pattern, length=length,
                         waist=waist, location=location)
        self.style = style
        self.jeans_version = 1

    @property
    def item_name(self):
```

```
        if StockItem.show_instrumentation:
            print('** Jeans get item_name called')
        return 'Jeans'

    def check_version(self):
        if StockItem.show_instrumentation:
            print('** Jeans check_version called')

        super().check_version()
        pass

    def __str__(self):
        if StockItem.show_instrumentation:
            print('** Jeans __str__ called')
        pants_details = super().__str__()
        template = '''{0}
Style: {1}'''
        return template.format(pants_details, self.style)


class Hat(StockItem):

    def __init__(self, stock_ref, price, color, size, location):
        if StockItem.show_instrumentation:
            print('** Hat __init__ called')
        super().__init__(stock_ref=stock_ref, price=price,
                         color=color, location=location)
        self.size = size
        self.Hat_version = 1

    @property
    def item_name(self):
        if StockItem.show_instrumentation:
            print('** Hat get item_name called')
        return 'Hat'

    def check_version(self):
        if StockItem.show_instrumentation:
            print('** Hat check_version called')

        super().check_version()
        pass

    def __str__(self):
```

```
        if StockItem.show_instrumentation:
            print('** Hat __str__ called')
        stock_details = super().__str__()
        template = '''{0}
Size: {1}'''
        return template.format(stock_details, self.size)

class Blouse(StockItem):

    def __init__(self, stock_ref, price, color, size, style, pattern, location):
        if StockItem.show_instrumentation:
            print('** Blouse __init__ called')
        super().__init__(stock_ref, price, color, location)
        self.size = size
        self.style = style
        self.pattern = pattern
        self.Hat_version = 1

    @property
    def item_name(self):
        if StockItem.show_instrumentation:
            print('** Blouse get item_name called')
        return 'Blouse'

    def check_version(self):
        if StockItem.show_instrumentation:
            print('** Blouse check_version called')

        super().check_version()
        pass

    def __str__(self):
        if StockItem.show_instrumentation:
            print('** Blouse __str__ called')
        stock_details = super().__str__()
        template = '''{0}
Size: {1}
Style: {2}
Pattern: {3}'''
        return template.format(stock_details, self.size,
                               self.style, self.pattern)


import pickle
```

```
class FashionShop:

    show_instrumentation = False

    def __init__(self):
        if FashionShop.show_instrumentation:
            print('** FashionShop __init__ called')
        self.__stock_dictionary = {}

    def save(self, filename):

        if FashionShop.show_instrumentation:
            print('** FashionShop save called')
        with open(filename,'wb') as out_file:
            pickle.dump(self,out_file)

    @staticmethod
    def load(filename):

        if FashionShop.show_instrumentation:
            print('** FashionShop load called')
        with open(filename,'rb') as input_file:
            result = pickle.load(input_file)


        for stock_item in result.__stock_dictionary.values():
            stock_item.check_version()
        return result


    def store_new_stock_item(self, stock_item):

        if FashionShop.show_instrumentation:
            print('** FashionShop store_new_stock_item called')
        if stock_item.stock_ref in self.__stock_dictionary:
            raise Exception('Item already present')
        self.__stock_dictionary[stock_item.stock_ref] = stock_item


    def find_stock_item(self, stock_ref):

        if FashionShop.show_instrumentation:
```

```
            print('** FashionShop find_stock_item called')
        if stock_ref in self.__stock_dictionary:
            return self.__stock_dictionary[stock_ref]
        else:
            return None

    def __str__(self):
        if FashionShop.show_instrumentation:
            print('** FashionShop __str__ called')
        stock = map(str,self.__stock_dictionary.values())
        stock_list = '\n'.join(stock)
        template = '''Items in Stock

{0}
'''
        return template.format(stock_list)


class FashionShopShellApplication:

    def __init__(self, filename):

        FashionShopShellApplication.__filename = filename
        try:
            self.__shop = FashionShop.load(filename)
        except:
            print('Fashion shop not loaded.')
            print('Creating an empty fashion shop')
            self.__shop = FashionShop()

    def create_new_stock_item(self):

        def get_stock_details():

            result = {}
            result['stock_ref'] = read_text('Enter stock reference: ')
            result['price'] = read_float_ranged(prompt='Enter price: ',
                                min_value=StockItem.min_price,
                                max_value=StockItem.max_price)
            result['color'] = read_text('Enter color: ')
            result['location'] = read_text('Enter location: ')
            return result

        menu = '''
```

Create new stock item

1: Dress
2: Pants
3: Hat
4: Blouse
5: Jeans

What kind of item do you want to add: '''

```
    item = read_int_ranged(prompt=menu,min_value=1,max_value=5)

    if item == 1:
       print('Creating a Dress')
       stock_details=get_stock_details()
       pattern = read_text('Enter pattern: ')
       size = read_text('Enter size: ')
       stock_item =  Dress(stock_ref=stock_details['stock_ref'],
                     price=stock_details['price'],
                     color=stock_details['color'],
                     location=stock_details['location'],
                     pattern=pattern,
                     size=size)
    elif item == 2:
       print('Creating a pair of Pants')
       stock_details=get_stock_details()
       pattern = read_text('Enter pattern: ')
       length = read_text('Enter length: ')
       waist = read_text('Enter waist: ')
       stock_item = Pants(stock_ref=stock_details['stock_ref'],
                      price=stock_details['price'],
                      color=stock_details['color'],
                      location=stock_details['location'],
                      pattern=pattern,
                      length=length,
                      waist=waist)
    elif item == 3:
       print('Creating a Hat')
       stock_details=get_stock_details()
       size = read_text('Enter size: ')
       stock_item = Hat(stock_ref=stock_details['stock_ref'],
                    price=stock_details['price'],
                    color=stock_details['color'],
                    location=stock_details['location'],
```

```
                    size=size)
    elif item == 4:
        print('Creating a Blouse')
        stock_details=get_stock_details()
        size = read_text('Enter size: ')
        style = read_text('Enter style: ')
        pattern = read_text('Enter pattern: ')
        stock_item = Blouse(stock_ref=stock_details['stock_ref'],
                            price=stock_details['price'],
                            color=stock_details['color'],
                            location=stock_details['location'],
                            pattern=pattern,
                            size=size,
                            style=style)
    elif item == 5:
        print('Creating some jeans')
        stock_details = get_stock_details()
        pattern = read_text('Enter pattern: ')
        length = read_text('Enter length: ')
        waist = read_text('Enter waist: ')
        style = read_text('Enter style: ')
        stock_item = Jeans(stock_ref=stock_details['stock_ref'],
                           price=stock_details['price'],
                           color=stock_details['color'],
                           location=stock_details['location'],
                           pattern=pattern,
                           length=length,
                           waist=waist,
                           style=style)

    try:
        self.__shop.store_new_stock_item(stock_item)
        print('Item stored')
    except Exception as e:
        print('Item not stored ')
        print(e)

def add_stock(self):

    print('Add stock')

    item_stock_ref = read_text('Enter the stock reference: ')

    item = self.__shop.find_stock_item(item_stock_ref)
```

```
    if item == None:
        print('This stock item was not found')
        return

    print(item)

    number_to_add = read_int_ranged('Number to add (0 to abandon): ',
                                    0, StockItem.max_stock_add)

    if number_to_add == 0:
        print('No items added')
    else:
        item.add_stock(number_to_add)
        print(item)

def sell_stock(self):

    print('Sell item')

    item_stock_ref = read_text('Enter the stock reference: ')

    item = self.__shop.find_stock_item(item_stock_ref)

    if item == None:
        print('This item was not found')
        return

    print('Selling')
    print(item)

    if item.stock_level == 0:
        print('There are none in stock')
        return

    number_sold = read_int_ranged('How many sold (0 to abandon): ',
                                  0,
                                  item.stock_level)

    if number_sold == 0:
        print('Sell item abandoned')
        return

    item.sell_stock(number_sold)
```

```
        print('Items sold')

    def do_report(self):
        print('Stock report')
        print(self.__shop)

    def main_menu(self):

        prompt = '''Mary's Fashion Shop

1: Create new stock item
2: Add stock to existing item
3: Sell stock
4: Stock report
5: Exit

Enter your command: '''

        while(True):
            command = read_int_ranged(prompt, 1, 5)
            if command == 1:
                self.create_new_stock_item()
            elif command == 2:
                self.add_stock()
            elif command == 3:
                self.sell_stock()
            elif command == 4:
                self.do_report()
            elif command == 5:
                self.__shop.save(FashionShopShellApplication.__filename)
                print('Shop data saved')
                break

ui = FashionShopShellApplication('dressshop1.pickle')
ui.main_menu()
```