Integrative Programming and Technologies

IT0011

Lladoc, Eadward Andrei D. ***TB22***

Activity 6: OOP and Exception

Program Problem

Create an item management application (CRUD application). The Item must consist of the following id, name, description and price. Provide a class that defines the item. Apply the necessary exception handling and validation to this program.

Answer:

Source Code: Screenshot

```
import uuid

class Item: # this will represent an item with a unique ID, name, description, and price
    def __init__(self, name: str, description: str, price: float):
        self.id = str(uuid.uuid4())  # this function generates a unique ID for every item
        self.name = self.validate_name(name)
        self.description = description
        self.price = self.validate_price(price)

    @staticmethod
    def validate_name(name): # this validates if the name is not a null string
        if not name or not isinstance(name, str):
            raise ValueError("Item name must not be an empty string.")
        return name.strip()

    @staticmethod
    def validate_price(price): # this validates if the price is a positive number
        if not isinstance(price, (int, float)) or price < 0:
            raise ValueError("Price must not be a negative number.")
        return round(price, 2)

    def __str__(self): # this returns a string representation of the item
        return f"ID: {self.id} | Name: {self.name} | Description: {self.description} | Price: ${self.price:.2f}"

class ItemManager: # this will manage the CRUD operations for tiems
    def __init__(self):
        self.items = {}

    def create_item(self, name, description, price):
        try: # this creates and adds a new item to the collection
            item = Item(name, description, price)
            self.items[item.id] = item
            print("Item added successfully!")
        except ValueError as e:
            print(f"Error: {e}") # error handling

    def read_items(self): # this will display all items in the collection
        if not self.items:
            print("No items available.")
        else:
            for item in self.items.values():
                print(item)

    def update_item(self, item_id, name=None, description=None, price=None):
        item = self.items.get(item_id) # updates an existing item's attributes
        if not item:
            print("Item not found.")
            return
        try:
            if name:
                item.name = Item.validate_name(name)
            if description:
                item.description = description
            if price is not None:
                item.price = Item.validate_price(price)
            print("Item updated successfully!")
        except ValueError as e:
            print(f"Error: {e}")
```

```
    def delete_item(self, item_id): # this deletes an item by ID
        if item_id in self.items:
            del self.items[item_id]
            print("Item deleted successfully!")
        else:
            print("Item not found.") # if the ID inputted does not exist

if __name__ == "__main__": # MAIN MENU yay
    manager = ItemManager()
    while True:
        print("\nEADWARD'S ONLINE INVENTORY MANAGEMENT SYSTEM MENU:")
        print("1. Add Item")
        print("2. View Items")
        print("3. Update Item")
        print("4. Delete Item")
        print("5. Exit")

        choice = input("Enter your choice: ") # prompts the user to input a value

        if choice == "1":
            name = input("Enter item name: ")
            description = input("Enter item description: ")
            try:
                price = float(input("Enter item price: "))
                manager.create_item(name, description, price)
            except ValueError:
                print("Invalid price. Please enter a numeric value.")
        elif choice == "2":
            manager.read_items()
        elif choice == "3":
            item_id = input("Enter item ID to update: ")
            name = input("Enter new name (leave blank to keep current): ") or None
            description = input("Enter new description (leave blank to keep current):
            price_input = input("Enter new price (leave blank to keep current): ")
            price = float(price_input) if price_input else None
            manager.update_item(item_id, name, description, price)
        elif choice == "4":
            item_id = input("Enter item ID to delete: ")
            manager.delete_item(item_id)
        elif choice == "5":
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please try again.")
```

Source Code: Copy & Paste

```
import uuid

class Item: # this will represent an item with a unique ID, name, description, and price
    def __init__(self, name: str, description: str, price: float):
        self.id = str(uuid.uuid4())  # this function generates a unique ID for every item
        self.name = self.validate_name(name)
        self.description = description
        self.price = self.validate_price(price)

    @staticmethod
    def validate_name(name): # this validates if the name is not a null string
        if not name or not isinstance(name, str):
            raise ValueError("Item name must not be an empty string.")
        return name.strip()

    @staticmethod
    def validate_price(price): # this validates if the price is a positive number
        if not isinstance(price, (int, float)) or price < 0:
            raise ValueError("Price must not be a negative number.")
        return round(price, 2)

    def __str__(self): # this returns a string representation of the item
```

```
        return f"ID: {self.id} | Name: {self.name} | Description: {self.description} | Price: ${self.price:.2f}"

class ItemManager: # this will manage the CRUD operations for tiems
    def __init__(self):
        self.items = {}

    def create_item(self, name, description, price):
        try: # this creates and adds a new item to the collection
            item = Item(name, description, price)
            self.items[item.id] = item
            print("Item added successfully!")
        except ValueError as e:
            print(f"Error: {e}") # error handling

    def read_items(self): # this will display all items in the collection
        if not self.items:
            print("No items available.")
        else:
            for item in self.items.values():
                print(item)

    def update_item(self, item_id, name=None, description=None, price=None):
        item = self.items.get(item_id) # updates an existing item's attributes
        if not item:
            print("Item not found.")
            return
        try:
            if name:
                item.name = Item.validate_name(name)
            if description:
                item.description = description
            if price is not None:
                item.price = Item.validate_price(price)
            print("Item updated successfully!")
        except ValueError as e:
            print(f"Error: {e}")

    def delete_item(self, item_id): # this deletes an item by ID
        if item_id in self.items:
            del self.items[item_id]
            print("Item deleted successfully!")
        else:
            print("Item not found.") # if the ID inputted does not exist

if __name__ == "__main__": # MAIN MENU yay
    manager = ItemManager()
    while True:
        print("\nEADWARD'S ONLINE INVENTORY MANAGEMENT SYSTEM MENU:")
        print("1. Add Item")
        print("2. View Items")
        print("3. Update Item")
        print("4. Delete Item")
        print("5. Exit")

        choice = input("Enter your choice: ") # prompts the user to input a value

        if choice == "1":
            name = input("Enter item name: ")
            description = input("Enter item description: ")
            try:
                price = float(input("Enter item price: "))
                manager.create_item(name, description, price)
```

```
            except ValueError:
                print("Invalid price. Please enter a numeric value.")
        elif choice == "2":
            manager.read_items()
        elif choice == "3":
            item_id = input("Enter item ID to update: ")
            name = input("Enter new name (leave blank to keep current): ") or None
            description = input("Enter new description (leave blank to keep current): ") or None
            price_input = input("Enter new price (leave blank to keep current): ")
            price = float(price_input) if price_input else None
            manager.update_item(item_id, name, description, price)
        elif choice == "4":
            item_id = input("Enter item ID to delete: ")
            manager.delete_item(item_id)
        elif choice == "5":
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please try again.")
```

Output:

```
EADWARD'S ONLINE INVENTORY MANAGEMENT SYSTEM MENU:
1. Add Item
2. View Items
3. Update Item
4. Delete Item
5. Exit
Enter your choice: 1
Enter item name: iPhone 11
Enter item description: Apple smartphone
Enter item price: 30000
Item added successfully!

EADWARD'S ONLINE INVENTORY MANAGEMENT SYSTEM MENU:
1. Add Item
2. View Items
3. Update Item
4. Delete Item
5. Exit
Enter your choice: 2
ID: be060b2e-9114-45b3-b3ae-eeadee1db316 | Name: iPhone 11 | Description: Apple
smartphone | Price: $30000.00

EADWARD'S ONLINE INVENTORY MANAGEMENT SYSTEM MENU:
1. Add Item
2. View Items
3. Update Item
4. Delete Item
5. Exit
Enter your choice: 3
Enter item ID to update: be060b2e-9114-45b3-b3ae-eeadee1db316
Enter new name (leave blank to keep current): Samsung A11
Enter new description (leave blank to keep current): Samsung smartphone
Enter new price (leave blank to keep current): 25000
Item updated successfully!

EADWARD'S ONLINE INVENTORY MANAGEMENT SYSTEM MENU:
1. Add Item
2. View Items
3. Update Item
4. Delete Item
```

```
Item updated successfully!

EADWARD'S ONLINE INVENTORY MANAGEMENT SYSTEM MENU:
1. Add Item
2. View Items
3. Update Item
4. Delete Item
5. Exit
Enter your choice: 2
ID: be060b2e-9114-45b3-b3ae-eeadee1db316 | Name: Samsung A11 | Description: Sams
ung smartphone | Price: $25000.00

EADWARD'S ONLINE INVENTORY MANAGEMENT SYSTEM MENU:
1. Add Item
2. View Items
3. Update Item
4. Delete Item
5. Exit
Enter your choice: 4
Enter item ID to delete: be060b2e-9114-45b3-b3ae-eeadee1db316
Item deleted successfully!

EADWARD'S ONLINE INVENTORY MANAGEMENT SYSTEM MENU:
1. Add Item
2. View Items
3. Update Item
4. Delete Item
5. Exit
Enter your choice: 2
No items available.

EADWARD'S ONLINE INVENTORY MANAGEMENT SYSTEM MENU:
1. Add Item
2. View Items
3. Update Item
4. Delete Item
5. Exit
Enter your choice: 5
Exiting...
```