

```
    p ngoại lệ cơ sở cho tất cả các lỗi trong hệ thống thư viện."""
    pass

class BookError(LibrarySystemError):
    """Lỗi liên quan đến các thao tác hoặc dữ liệu của Book."""
    pass

class InvalidBookDataError(BookError):
    """Được ném ra khi cố gắng tạo Book với dữ liệu không hợp lệ."""
    pass

class BookAlreadyBorrowedError(BookError):
    """Được ném ra khi cố gắng mượn một cuốn sách đã được mượn."""
    pass

class BookAlreadyAvailableError(BookError):
    """Được ném ra khi cố gắng trả một cuốn sách đang có sẵn."""
    pass
```

```
        cho 'price'
@property
def price(self) -> float:
    """Trả về giá của sách."""
    return self._price
```

```
1 #
2 # --- TRIỂN KHAI LỚP LIBRARY ---
3 #
4
5 from typing import List, Dict, Optional, Any
6
7 class Library:
8     """
9     Quản lý một tập hợp các đối tượng Book.
10    """
11
12    def __init__(self, name: str):
13        """
14        Khởi tạo một thư viện mới.
```

```
ong thư viện: {a  
book.display :  
def get_sto
```

mu

```

library.add_book(book1)
library.add_book(book2)
library.add_book(book3)

# 3. Hiển thị tất cả sách
library.display_all_books()

# 4. Muốn sách
print("\n")
book_to_borrow = library.find_book("978-02")
if book_to_borrow:
    book_to_borrow.borrow()

# 5. Thống kê
print()
stats = library.get_statistics()
print(stats) # {'total_books': 3, 'available_books': 2, 'borrowed_books': 1}

# 6. Trả sách
print("\n")
if book_to_borrow:
    book_to_borrow.return_book()

# 7. Thống kê lại
print()
stats_after_return = library.get_statistics()
print(stats_after_return) # {'total_books': 3, 'available_books': 3, 'borrowed_books': 0}

except Exception as e:
    # Bắt bất kỳ lỗi không mong muốn nào khác
    print(f"\n--- ĐÃ XÁY RA LỖI HỆ THỐNG KHÔNG MONG MUỐN ---")
    print(f"Loi: {type(e).__name__} - {e}")

```

```

1 # Chạy demo
2 run_library_demo()

```

```

--- Bắt đầu Demo Hệ thống Quản lý Thư viện ---

Đã tạo thư viện: Thư viện Quốc gia
Đã thêm sách: 'Lập trình Python' vào thư viện Thư viện Quốc gia.
Đã thêm sách: 'Giải tích 1' vào thư viện Thư viện Quốc gia.
=====

Toàn bộ sách tr
```

14

```

        iết kế OOP
Tác giả: Alan Turing
ISBN: 978-03
Giá: 185,000.00đ
Trạng thái: Có sẵn
-----
```

Đã mượn sách: 'Cầu t

f

```
te]: Xác thực email PayPal THẤT BẠI. (Email không
return False

def process_payment(self):
    print(f"    [Process]: Đang chuyển hướng đến PayPal API để xử lý ${sel
    print(f"    [Process]: Xử lý PayPal HOÀN TẮT.")
```

```
1 class BankTransferPayment(PaymentMethod):
2     """
3     Triển khai cụ thể cho thanh toán bằng Chuyển khoản Ngân hàng.
4     """
5     def __init__(self, amount, description, account_number, bank_code):
6         super().__init__(amount, description)
7         self.account_number = account_number
8         self.bank_code = bank_code
9         print(f" -> Đối tượng BankTransferPayment được tạo (Ngân hàng: {self.ban
10
11     def validate(self):
12         """
13         và
14
15
16
17
18         nt("    [Validate]: Xác thực tài khoản ngân hàng THẤT BẠI.")
19         return False
20
21     def process_payment(self):
22         print(f"    [Process]: Đang tạo yêu cầu chuyển khoản ${self.amount:.2f} qua h
23         print(f"    [Process]: Xử lý chuyển khoản HOÀN TẮT.")
24
25 class Digita
```

5  
7  
3  
29d

```
ich '{payment_method.description}' BỊ HỦY.")  
  
# 5. Lưu kết quả vào lịch sử  
self.transaction_history.append({  
    "description": payment_method.description,  
    "amount": payment_method.amount,  
    "type": type(pay)
```



1



```
---  
Giao dịch:  
- Mô tả:  
- Số tiền:  
- Loại:  
- Trạng thái:  
---  
Giao dịch 5:  
- Mô tả:  
- Số tiền:  
- Loại:  
- Trạng thái:  
---  
Giao dịch 6:  
- Mô tả: Thanh toán  
- Số tiền: $250.00  
- Loại: Bank  
- Trạng thái: THÁT  
---  
Giao dịch 7:  
- Mô tả: Nạp điện  
- Số tiền: $120000  
- Loại: Digital  
- Trạng thái: THANH TOÁN  
---  
Giao dịch 8:  
- Mô tả: Mua nước  
- Số tiền: $50000.00  
- Loại: Digital  
- Trạng thái: THÁT BẠI (>)  
---  
=====
```

## ▼ BÀI TẬP VỀ NHÀ

### ▼ Bài 1 – Hệ thống Quản lý Sinh viên

```
1 import datetime  
2  
3 # =====  
4 # ĐỊNH NGHĨA  
5  
6  
7  
8  
9  
10  
11 """  
12     def __init__(self, student_id, name, email, birth_year):  
13         self.student_id = student_id  
14         self.name = name  
15         self.email = email  
16         self.birth_year = birth_year  
17         self.enrollments = []# Danh sách các đối tượng Enrollment  
18  
19     def calculate_age(self):  
20         """Tính tuổi của sinh viên  
21  
22             self.birth_year  
23  
24     def display_info(self):  
25         """Hiển thị thông tin chi tiết của sinh viên."""  
26         print(f"--- Thông tin sinh viên ---")  
27         print(f"Mã SV: {self.student_id}")  
28         print(f"Họ tên: {self.name}")  
29         print(f"Email: {self.email}")  
30         print(f"Năm sinh: {self.birth_year}")  
31         print(f"Tuổi: {self.calculate_age()}")  
32         print(f"Số môn đã đăng ký: {len(self.enrollments)}")  
33  
34     def add_enrollment(self, enrollment):  
35         """Thêm một lượt đăng ký môn học cho sinh viên."""
```

```
36     self.enrollments.append(enrollment)
37
38     def calculate_gpa(self):
39         """
40             Tính điểm GPA của sinh viên.
41             GPA = Tổng (điểm trung bình môn * số tín chỉ) / Tổng số tín chỉ
42         """
43         total_weighted_score = 0
44         total_credits = 0
45
46         for enrollment in self.enrollments:
47             grade = enrollment.calculate_final_score()
48             if grade is not None: # Chỉ tính GPA cho các môn đã có điểm
49                 credits = enrollment.course.credits
50                 total_weighted_score += grade * credits
51                 total_credits += credits
52
53         if total_credits == 0:
54             return 0.0
55
56         return total_weighted_score / total_credits
57
58     def __str__(self):
59         return f"{self.name} ({self.student_id})"
```

```
1 class Course:
2     """
3         Lớp đại diện cho một khóa học.
4         Lưu trữ tên khóa học, số tín chỉ và danh sách sinh viên đăng ký.
5     """
6     def __init__(self, course_id, name, credits, max_students=5):
7
8         self.course_id = course_id
9         self.name = name
10        self.credits = credits
11        self.max_students = max_students
12        self.students = []# Danh sách các đối tượng sinh viên
```



```
39         return self
40
41     def register(self, name):
42         """Registers a new component with the component manager.
43
44         :param str name: The name of the component to register.
45
46         :return: The registered component object.
47
48     """
49
50     def __init__(self, name):
51         self.name = name
52
53         self._components = {}
54
55         self._register()
56
57     def _register(self):
58         self._components[self.name] = self
59
60
61     def __str__(self):
62         return f"Component Manager: {self.name}
```

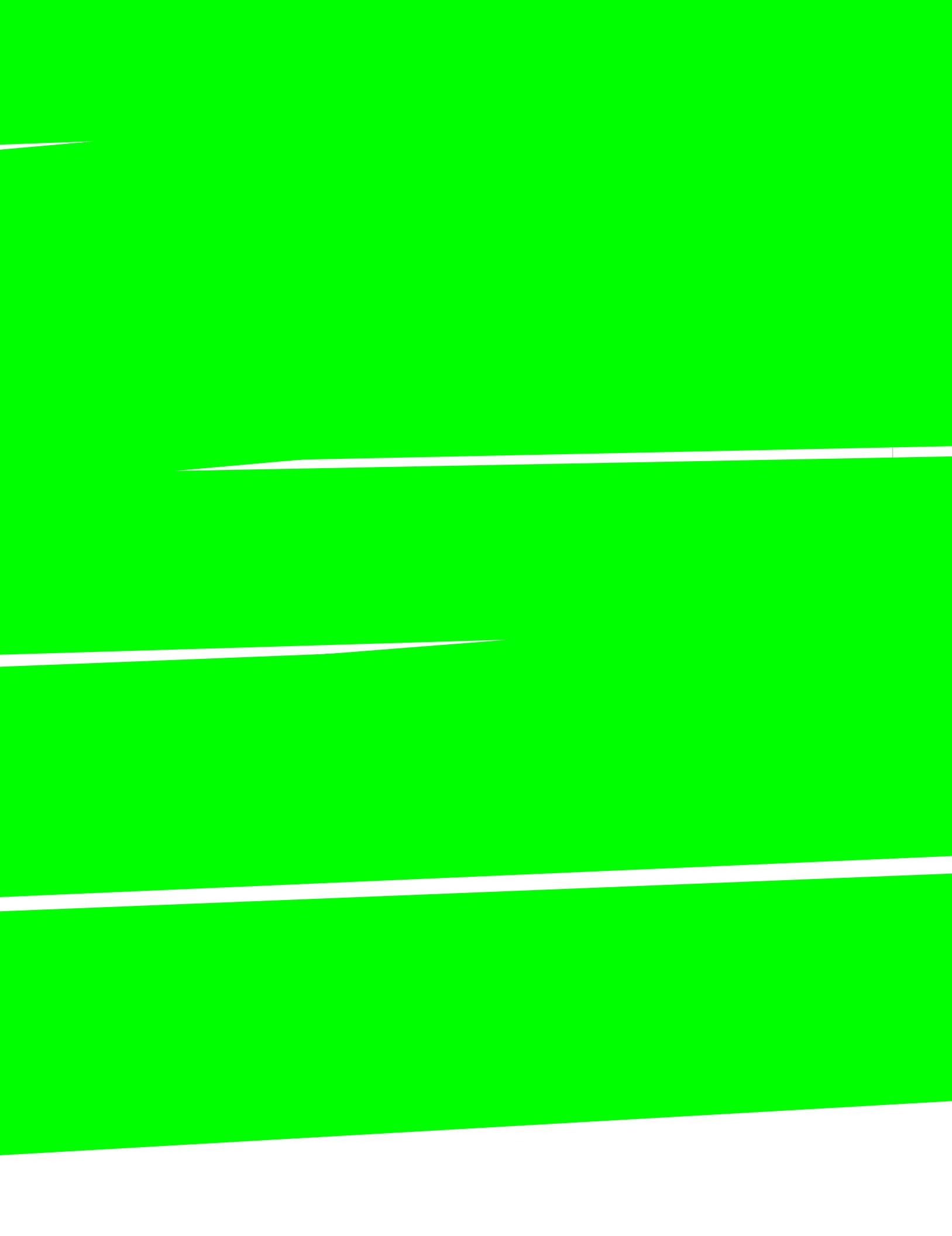
```
66
67
68
69
70
71
```

```
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
```

```
125         print(f"Không tìm thấy sinh viên nào với từ khóa '{keyword}'.")
126     else:
127         print(f"Tìm thấy {len(results)} sinh viên:")
128         for student in results:
129             student.display_info()
130     return results
131
132 def list_all_students(self):
133     """Tiện ích: Liệt kê tất cả sinh viên."""
134     if not self.students:
135         print("Chưa có sinh viên nào trong hệ thống.")
136     return
137     print("--- DANH SÁCH SINH VIÊN ---")
138     for student in self.students.values():
139         print(f"- {student.name} (ID: {student.student_id})")
140
141 def list_all_courses(self):
142     """Tiện ích: Liệt kê tất cả môn học."""
143     if not self.courses:
144         print("Chưa có môn học nào trong hệ thống.")
145     return
146     print("--- DANH SÁCH MÔN HỌC ---")
147     for course in self.courses.values():
148         print(f"- {course.name} (ID: {course.course_id}, Tín chỉ: {course.credits})")
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37         print("Nhập thông tin sinh viên")
38         print("1. Thêm sinh viên")
39         print("2. Sửa thông tin sinh viên")
40         print("3. Xóa thông tin sinh viên")
41         print("4. Tính trung bình môn học")
42         print("5. Xem danh sách sinh viên")
43         print("6. Tạo mới danh sách thi tất cả sinh viên")
44         print("7. Xem danh sách thi tất cả môn học")
45         print("8. Thoát")
46
47         choice = input("Vui lòng chọn chức năng (0-10): ")
48
49         if choice == '1':
50             # Thêm sinh viên
51             sid = input("Nhập mã SV: ")
52             name = input("Nhập tên: ")
53             email = input("Nhập email: ")
54             try:
55                 year = int(input("Nhập năm sinh: "))
56                 system.add_student(sid, name, email, year)
57             except ValueError:
58                 print("Lỗi: Năm sinh phải là một con số.")
59
60         elif choice == '2':
61             # Sửa thông tin sinh viên
62             print("Chưa có chức năng này")
63
64         elif choice == '3':
65             # Xóa thông tin sinh viên
66             print("Chưa có chức năng này")
67
68         elif choice == '4':
69             # Tính trung bình môn học
70             print("Chưa có chức năng này")
71
72         elif choice == '5':
73             # Xem danh sách sinh viên
74             print("Danh sách sinh viên:")
75             print(system.get_all_students())
76
77         elif choice == '6':
78             # Tạo mới danh sách thi tất cả sinh viên
79             print("Chưa có chức năng này")
80
81         elif choice == '7':
82             # Xem danh sách thi tất cả môn học
83             print("Chưa có chức năng này")
84
85         elif choice == '8':
86             # Thoát
87             print("Xin chào! Đến hẹn giờ")
88             break
89
90         else:
91             print("Chọn sai chức năng")
```

```
62
63
64
65
66         nhập số tối đa: "))
67     system.add_course(cid, name, credits, max_std)
68 except ValueError:
69     print("Lỗi: Tín chỉ và số phải là con số.")
70
71 elif choice == '3':
72     # Đăng ký môn học
73     sid = input("Nhập mã SV cần đăng ký: ")
74     cid = input("Nhập mã môn học: ")
75     system.register_student_to_course(sid, cid)
76
77
78
79         nhập mã SV cần hủy đăng ký: ")
80 cid = input("Nhập mã môn học: ")
81 system.drop_student_from_course(sid, cid)
82
83 elif c
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
```



```
27     if amount > self.balance:
28         # Thiết lập một "hợp đồng" (contract): ném ra lỗi nếu giao dịch thất bại
29         raise ValueError("Không đủ số dư để thực hiện giao dịch.") [12, 13]
30
31     self.balance -= amount
32     # Ghi lại giao dịch sau khi thành công
33     self.transactions.append(Transaction("Rút tiền", amount, self.balance))
34     print(f'Rút tiền thành công {amount:.0f} đ. Số dư mới: {self.bala
35
36
37
38
39
40
41
42
43
44
45
46
47
48
```

```

21         raise ValueError(f"Không đủ hạn mức thấu chi. (Yêu cầu {total_to_deduct:.0f} đ, chỉ có {available_to_withdraw:.0f} đ)")
22
23     # Logics của cha không còn áp dụng. Lớp con thay thế hoàn toàn.
24     self.balance -= total_to_deduct
25     # Ghi lại giao dịch Rút tiền và Phí
26     self.transactions.append(Transaction(f"Rút tiền (Phí {self.transaction_fee:.0f} đ)", amount, self.balance))
27     print(f"Rút tiền thành công {amount:.0f} đ (Phí {self.transaction_fee:.0f} đ). Số dư mới: {self.balance:.0f} đ.")
28     return True

```

```

1 def transfer(from_account, to_account, amount):
2     print(f"\n--- Thực hiện chuyển {amount:.0f} đ từ {from_account.owner_name} đến {to_account.owner_name} ---")
3
4     # Kiểm tra tính hợp lệ của đối tượng
5     if not isinstance(from_account, BankAccount) or not isinstance(to_account, BankAccount):
6         print("Lỗi: Tài khoản không hợp lệ.")
7         return False
8
9     try:
10         # 1. Thử rút tiền từ tài khoản nguồn
11         if from_account.withdraw(amount):
12             # 2. Chỉ gửi tiền nếu rút thành công
13             to_account.deposit(amount)
14             print("Chuyển khoản thành công.")
15             return True
16         else:
17             # Bắt các trường hợp withdraw trả về False (ví dụ: SavingsAccount hết lượt)
18             print("Chuyển khoản thất bại (Rút tiền bị từ chối, có thể do hết giới hạn.)")
19             return False
20
21     except ValueError as e:
22         # Bắt lỗi từ BankAccount.withdraw hoặc CheckingAccount.withdraw
23
24
25
26
27
28     return False

```

```

1 #####
2 # GIAO DỊCH
3 #####
4 class Transaction:
5     """
6     Ghi lại một giao dịch đơn lẻ, bao gồm thời gian, loại, số tiền,
7     và số dư sau giao dịch.
8     """
9     def __init__(self, trans_type, amount, balance_after):
10         self.trans_type = trans_type
11         self.amount = amount
12         self.timestamp = datetime.now()
13         self.balance_after = balance_after
14
15     def __str__(self):
16         # Tự định dạng đối tượng khi được in
17         # Định dạng thời gian: DD/MM/YYYY HH:MM:SS
18         formatted_time = self.timestamp.strftime('%d/%m/%Y %H:%M:%S')
19         return f"[{formatted_time}] {self.trans_type}:<10"

```

```

1 #####
2 # BANKACCOUNT
3 #####
4 class BankAccount:
5     """
6     Lớp cơ sở quản lý các thuộc tính và hành vi chung của một tài khoản ngân hàng.
7     """
8     def __init__(self, owner_name, initial_balance=0):
9         # Sinh ID ngẫu nhiên
10        self.account_number = str(uuid.uuid4().hex[:10].upper())
11        self.owner_name = owner_name
12        self.balance = initial_balance
13        self.transactions = []# Lưu trữ lịch sử giao dịch
14        print(f"Đã mở tài khoản {self.account_number} cho {self.owner_name} với số dư {self.balance:.0f} đ.")
15
16    def deposit(self, amount):
17        """Gửi một số tiền vào tài khoản."""
18        if amount <= 0:
19            print("Lỗi: Số tiền gửi phải lớn hơn 0.")
20            return False
21
22        self.balance += amount
23        # Ghi lại mọi giao dịch
24        self.transactions.append(Transaction("Gửi tiền", amount, self.balance))
25        print(f"Gửi tiền thành công {amount:.0f} đ. Số dư mới: {self.balance:.0f} đ.")

```

```
        t, self.balance))
print(f"Đã cộng lãi {interest:.0f} đ. Số dư mới: {self.balance:.0f} đ.")

# GHI ĐỀ PHƯƠNG THỨC
def withdraw(self, amount):
    """
    Ghi đè 'withdraw' để thêm logic kiểm tra giới hạn rút tiền.
    """
    # 1. Logic riêng: Kiểm tra giới hạn
    if self.withdraw_count >= self.withdraw_limit:
        print(f"Lỗi: Vượt quá giới hạn {self.withdraw_limit} lần rút tiền.")
        return False # Lỗi nghiệp vụ, trả về False

    # 2. Logic cha: Kiểm tra số dư (bằng cách gọi 'super()')
    try:
        success = super().withdraw(amount) # Thủ rút tiền bằng logic của cha
        if success:
            self.withdraw_count += 1 # Tăng bộ đếm nếu thành công
            print(f"(Còn {self.withdraw_limit - self.withdraw_count} lượt rút)")
            return True
        return False
    except ValueError as e:
        # Bắt lỗi 'ValueError' từ lớp cha
        print(f'Lỗi giao dịch: {e}')
        return False
```

```

1 ##### CHECKINGACCOUNT (KẾ THỪA TỪ BANKACCOUNT)
2 # CHECKINGACCOUNT (KẾ THỪA TỪ BANKACCOUNT)
3 #####
4 class CheckingAccount(BankAccount):
5     """
6     Lớp tài khoản thanh toán, bổ sung thao chi và phí giao dịch.
7     Kế thừa từ BankAccount.
8     """
9     def __init__(self, owner_name, initial_balance=0, overdraft_limit=500000, transaction_fee=5000):
10         super().__init__(owner_name, initial_balance)
11         self.overdraft_limit = overdraft_limit # Hạn mức thao chi [23]
12         self.transaction_fee = transaction_fee # Phí rút tiền [24]
13
14     # GHI ĐỀ PHƯƠNG THỨC
15     def withdraw(self, amount):
16         """
17             Ghi đè 'withdraw' để thêm logic thao chi và phí giao dịch.
18             Phương thức này THAY THẾ hoàn toàn logic của cha.
19             """
20         if amount <= 0:
21             print("Lỗi: Số tiền rút phải lớn hơn 0.")
22             return False
23
24         # Tổng tiền bị trừ = số tiền rút + phí
25         total_to_deduct = amount + self.transaction_fee
26
27         # Hạn mức có thể rút = số dư + thao chi
28         available_to_withdraw = self.balance + self.overdraft_limit
29
30         if total_to_deduct > available_to_withdraw:
31             # Tuân thủ "hợp đồng" của lớp cha: ném lỗi nếu thất bại
32             raise ValueError(f"Không đủ hạn mức thao chi. (Yêu cầu {total_to_deduct:.0f} đ, có sẵn {available_to_withdraw:.0f} đ)")
33
34         # Thay thế logic của cha, cho phép số dư âm
35         self.balance -= total_to_deduct
36         self.transactions.append(Transaction(f'Rút tiền (Phí {self.transaction_fee:.0f} đ)", amount, self.balance)))
37         print(f'Rút tiền thành công {amount:.0f} đ (Phí {self.transaction_fee:.0f} đ). Số dư mới: {self.balance:.0f} đ.')
38         return True

```

```

1 ##### TRANSFER (CHUYỂN KHOẢN)
2 # TRANSFER (CHUYỂN KHOẢN)
3 #####
4 def transfer(from_account, to_account, amount):
5     """
6     Chuyển tiền giữa hai đối tượng tài khoản.
7     Sử dụng withdraw() và deposit().
8     """
9     print("\n--> Thực hiện chuyển {amount:.0f} đ từ {from_account.owner_name} đến {to_account.owner_name} -->")
10
11    if not isinstance(from_account, BankAccount) or not isinstance(to_account, BankAccount):
12        print("Lỗi: Tài khoản không hợp lệ.")
13        return False
14
15    try:
16        # 1. Thử rút tiền
17        from_account.withdraw(amount)
18        # 2. Nếu thành công, ném lỗi ValueError
19        raise ValueError('ValueError')
20
21        # 3. Nếu lỗi ValueError, ném lỗi TransferFailed
22        raise TransferFailed(f'Trích tiền thất bại: {from_account.owner_name} ({from_account.balance:.0f} đ)')
23
24        # 4. Nếu không lỗi ValueError, ném lỗi TransferSuccess
25        raise TransferSuccess(f'Trích tiền thành công: {from_account.owner_name} ({from_account.balance:.0f} đ) -> {to_account.owner_name} ({to_account.balance:.0f} đ)')
26
27        # 5. Nếu không lỗi TransferSuccess, ném lỗi TransferIncomplete
28        raise TransferIncomplete(f'Trích tiền không hoàn thành: {from_account.owner_name} ({from_account.balance:.0f} đ) -> {to_account.owner_name} ({to_account.balance:.0f} đ)')
29
30        # 6. Nếu không lỗi TransferIncomplete, ném lỗi TransferAborted
31        raise TransferAborted(f'Trích tiền bị hủy: {from_account.owner_name} ({from_account.balance:.0f} đ) -> {to_account.owner_name} ({to_account.balance:.0f} đ)')
32
33    finally:
34        # 7. Cuối cùng, ném lỗi TransferFinalFailure
35        raise TransferFinalFailure(f'Trích tiền cuối cùng thất bại: {from_account.owner_name} ({from_account.balance:.0f} đ) -> {to_account.owner_name} ({to_account.balance:.0f} đ)')

```

n 980,000 đ)

```
1 print("====")
2 print("CHỨC NĂNG 3: CHUYỂN TIỀN")
3
4
5         _acc, 500000)
6
7 # Chuyển thất bại (không đủ tiền)
8 # $_acc=10,000,000 (10M + 700k lãi - 1.7M rút + 500k nhận)
```

```
        sau khi bán"""
if quantity_sold <= self.stock:
    self.stock -= quantity_sold
    return True
else:
    print(f'Lỗi: Không đủ hàng tồn kho cho {self.name}, Chỉ còn {self.stock}.')
    return False
```

```
1. Khi lập trình Python
2
3
4     __init__(product_id, name, price, stock)
5     self.brand = brand
6     self.warranty_months = warranty_months
7
8 def display_info(self):
9     super().display_info()
```

```
1 class Clothing(Product):
2     def __init__(self, product_id, name, price, stock,
3
4
5
6
7
8
9         size}")
10    print(f"  Mau: {self.color}")
```

```
1 class Book(Product):
2     def __init__(self, product_id, name, price, stock, author, publisher):
3         super().__init__(product_id, name, price, stock)
4         self.author = author
5         self.publisher =
6
7
8
9
10
```

```
1 # LỚP GIỎ HÀNG (SHOPPING CART)
2 class ShoppingCart:
3     def __init__(self, customer):
4         self.customer = customer
5         # Dùng dictionary để lưu: {product_id: {'product': ProductObject, 'quantity': 2}}
6         self.items = {}
7
8     def add_product(self, product, quantity):
9         """Thêm / Cập nhật sản phẩm vào giỏ (Hướng dẫn 2)"""
10        if quantity <= 0:
11            print("Số lượng phải lớn hơn 0")
12            return
13
14        # Kiểm tra tồn kho
15        current_in_cart = self.items.get(product.product_id, {}).get('quantity', 0)
16        total_needed = current_in_cart + quantity
17
18        if total_needed > product.stock:
19            print(f'Lỗi: Không đủ hàng')
```

```
87     final_total = max(0, total - discount)
88     print(f'TỔNG CỘNG: {final_total} VNĐ')
89
90     # 5. Cập nhật tồn kho
91     print(f'Cập nhật tồn kho')
92
93     # 6. Cập nhật điểm tích lũy
94     print(f'ĐIỂM TÍCH LŨY: {customer.earned} VNĐ')
95     customer.add_points(earned)
96
97     # 7. Tạo đơn hàng mới
98     items_copy = items
99     new_order = Order(items_copy)
100
101    # 8. Giao hàng
102    print(f'GIAO HÀNG')
103    items = []
104
105    # --- THANH TOÁN ---
106    new_order
```

ông  
# Thành công  
# Thành công

```
s_to_redeem=100000)
order2.display_order()

# Kiểm tra điểm của khách
```

```
print("\n--- Kiểm tra sau Đơn 2 ---")
customer1.display_info() # Phải còn 188,5
```

```
ĐIỂM THƯỞNG) ======
Đã thêm 3 Áo Sơ Mi Trắng vào giỏ.
--- Giỏ hàng của Trần Văn An ---
* Áo Sơ Mi Trắng (x3) - Đơn giá: 500,000 VND
-----
Tạm tính: 1,500,000 VND
```

```
--- BẮT ĐẦU THANH TOÁN ---
```

```
Tổng tạm tính: 1,500,000 VND
Tiền sau khi giảm giá coupon: 1,500,000 VND
Áp dụng 100,000 điểm thưởng...
Thông báo: Đã dùng 100,000 điểm. Điểm còn lại: 174,500
TỔNG CỘNG PHẢI TRẢ: 1,400,000 VND
Cập nhật tồn kho...
Thông báo: Bạn được cộng 14,000 điểm.
--- THANH TOÁN THÀNH CÔNG ---
```

```
--- CHI TIẾT ĐƠN HÀNG: DH0002 ---
```

```
Ngày đặt: 2025-11-06
Khách hàng: Trần Văn An
Trạng thái: Pending
Sản phẩm đã mua:
* Áo Sơ Mi Trắng (x3)
TỔNG THANH TOÁN: 1,400,000 VND
-----
```

```
--- Kiểm tra sau Đơn 2 ---
```

```
Khách hàng: Trần Văn An (ID: KH001)
Điểm tích lũy: 188,500 điểm
```

```
1 print("===== CẬP NHẬT TRẠNG THÁI ĐƠN HÀNG =====")
2 # Cập nhật trạng thái cho đơn hàng 1
3 order1.display_order() # Trạng thái "Pending"
4 order1.update_status("Delivered")
5 order1.display_order() # Trạng thái "Delivered"
6
7 # Thử cập nhật trạng thái không hợp lệ
8 order2.update_status("Done") # Báo lỗi
```

```
===== CẬP NHẬT TRẠNG THÁI ĐƠN HÀNG =====
```

```
--- CHI TIẾT ĐƠN HÀNG: DH0001 ---
```

```
Ngày đặt: 2025-11-06
Khách hàng: Trần Văn An
Trạng thái: Pending
Sản phẩm đã mua:
```

