

**Session 12****Exceptions****Phần I - Thực hiện trong 120 phút****1.1 Mục tiêu**

- ✓ Hiểu vững khái niệm về quản lý lỗi trong Java.
- ✓ Hiểu rõ các dạng lỗi: do code logic, do hệ thống, do người dùng, do lập trình viên...v...v.
- ✓ Hiểu vững cú pháp khai báo lỗi try-catch-finally
- ✓ Hiểu cơ bản về cú pháp try-with-resource mới từ JDK 7.
- ✓ Hiểu rõ việc bố trí sắp xếp thứ tự để kiểm soát tối đa lỗi.
- ✓ Biết kế thừa lớp Exception để tạo những lớp kiểm soát lỗi cụ thể cho từng bài toán.

**1.2 Thực hiện**

**Bài thực hành 1:** Viết một chương trình Java kiểm soát việc thêm dữ liệu vào mảng sinh viên, nếu đã có thì thay thế còn nếu đạt tới giới hạn thì thực hiện mở rộng mảng trước khi thêm mới.

**Bước 1:** Viết một class tên là SinhVienManagement

```
package demo.jp1.lab10.baithuchanh1;

import java.util.Arrays;

/**
 *
 * @author minhvuvc
 */
public class SinhVienManagement {

    String[] arrSinhVien;

    public SinhVienManagement() {
        arrSinhVien = new String[0]; // Khởi tạo mảng 0 phần tử
    }

    //Bước 2: Viết hàm mở rộng dữ liệu
    private void moRong() {
```

```
int size = arrSinhVien.length + 1;
String[] temp = Arrays.copyOf(arrSinhVien, size);
arrSinhVien = new String[size];
System.arraycopy(temp, 0, arrSinhVien, 0, size);
}

//Bước 3: Viết các hàm thêm dữ liệu và kiểm soát ngoại lệ khi thêm
public void themSinhVien(String name, int pos) {
    try {
        arrSinhVien[pos - 1] = name;
    } catch (ArrayIndexOutOfBoundsException e) {
        moRong();
        arrSinhVien[arrSinhVien.length - 1] = name;
        // System.out.println("Lỗi: " + e);
    }
}

//Bước 4: Viết hàm hiển thị dữ liệu
public void hienThi() {
    for (int i = 0; i < arrSinhVien.length; i++) {
        String arrSinhVien1 = arrSinhVien[i];
        System.out.println(arrSinhVien1);
    }
}
}
```

**Bước 5:** Viết class tên MainClass có hàm main để kiểm thử

```
public class MainClass {
    /**
     * @param args thecommandlinearguments
     */
    public static void main(String[] args) {
        SinhVienManagement sv = new SinhVienManagement();
        sv.themSinhVien("Minh Dai Ka", 6);
        sv.hienThi();
    }
}
```

**Bước 6:** Chạy thử chương trình

Output - demo.jp1.lab10 (run)

```
run:
Minh Dai Ka
BUILD SUCCESSFUL (total time: 0 seconds)
```

**Bài thực hành 2:** Viết một chương trình Java mô phỏng hệ thống tài khoản ngân hàng, khách hàng có thể gửi tiền và rút tiền, yêu cầu số tiền trong tài khoản phải lớn hơn hoặc bằng số tiền rút.

**Bước 1:** Viết class `InsufficientFundsException` kế thừa lớp `Exception` (đổi tên lớp)

```
package demo.jp1.lab10.baithuchanh2;

/**
 *
 * @authorminhvuvc
 */
public class InsufficientFundsException extends Exception {

    private double amount;

    @Override
    public void printStackTrace() {
        System.err.println("Tài khoản của quý khách không đủ thực hiện giao
        dịch này!");
    }

    super.printStackTrace();

    public InsufficientFundsException(double amount) {
        this.amount = amount;
    }
}
```

```
public double getAmount() {  
    return amount;  
}  
}
```

**Bước 2:** Viết class CheckingAccount có các hàm thực hiện gửi tiền, rút tiền.

```
package demo.jp1.lab10.baithuchanh2;  
  
/**  
 *  
 * @authorminhvufc  
 */  
public class CheckingAccount {  
    private double balance;  
    private int number;  
    public CheckingAccount(int number) {  
        this.number = number;  
    }  
    public void deposit(double amount) {  
        balance += amount;  
    }  
    public void withdraw(double amount) throws InsufficientFundsException {  
        if (amount <= balance) {  
            balance -= amount;  
        } else {  
            double needs = amount - balance;  
            throw new InsufficientFundsException(needs);  
        }  
    }  
    public double getBalance() {  
        return balance;  
    }  
}
```

```
public int getNumber() {  
    return number;  
}  
}
```

**Bước 3:** Viết class BankDemo nhập một số dữ liệu cho CheckingAccount

```
package demo.jp1.lab10.baithuchanh2;  
  
/**  
 *  
 * @authorminhvufc  
 */  
public class BankDemo {  
    public static void main(String[] args) {  
        CheckingAccount c = new CheckingAccount(101);  
        System.out.println("Depositing $500...");  
        c.deposit(500.00);  
        try {  
            System.out.println("\nWithdrawing $100...");  
            c.withdraw(100.00);  
            System.out.println("\nWithdrawing $600...");  
            c.withdraw(600.00);  
        } catch (InsufficientFundsException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

**Bước 4:** Chạy kiểm thử chương trình

```
Output - demo.jp1.lab10 (run)
run:
Depositing $500...
Withdrawing $100...
Withdrawing $600...
Tài khoản của quý khách không đủ thực hiện giao dịch này!
demo.jp1.lab10.baithuchanh2.InsufficientFundsException
|   at demo.jp1.lab10.baithuchanh2.CheckingAccount.withdraw(CheckingAccount.java:30)
|   at demo.jp1.lab10.baithuchanh2.BankDemo.main(BankDemo.java:22)
BUILD SUCCESSFUL (total time: 0 seconds)
```

**Bài thực hành 3:** Viết một chương trình Java yêu cầu nhập dữ liệu 2 số kiểu float từ bàn phím. Thực hiện phép tính chia. Yêu cầu bắt lỗi khi nhập liệu nếu không phải kiểu số thì bắt lỗi `InputMismatchException`, nếu nhập số chia là 0 thì bắt lỗi `ArithmeticException`, kết thúc nếu ko có lỗi thì hiển thị "Dữ liệu nhập an toàn!".

**Bước 1:** Viết class `BaiThucHanh03` có hàm `main`

```
package demo.jp1.lab10.baithuchanh;

/**
 *
 * @author minhvuvc
 */
public class BaiThucHanh03 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }
}
```

**Bước 3:** Viết mã nhập liệu và bắt lỗi nhập liệu.

```
public static void main(String[] args) {
    try {
        Scanner nhap = new Scanner(System.in);
        int soA = nhap.nextInt();
        int soB = nhap.nextInt();
    } catch (InputMismatchException ime) {
        System.out.println("Dữ liệu nhập không hợp lệ - " + ime.toString());
    }
}
```

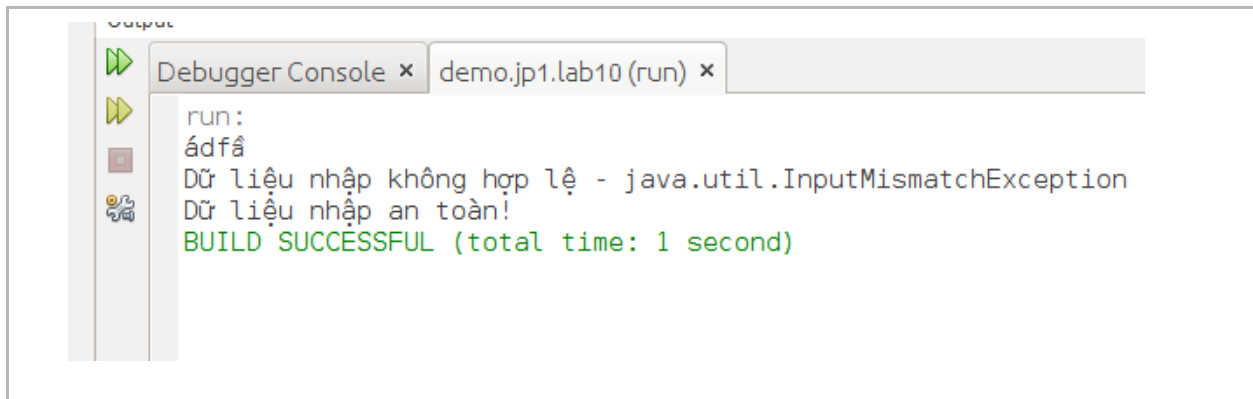
**Bước 3:** Viết mã tính toán chia 2 số và bắt lỗi tính toán

```
public static void main(String[] args) {  
    try {  
        Scanner nhap = new Scanner(System.in);  
        int soA = nhap.nextInt();  
        int soB = nhap.nextInt();  
  
        try {  
            System.out.println("Phép chia = " + (soA / soB));  
        } catch (ArithmeticException ae) {  
            System.out.println("Lỗi tính toán - " + ae.toString());  
        }  
    } catch (InputMismatchException ime) {  
        System.out.println("Dữ liệu nhập không hợp lệ - " + ime.toString());  
    }  
}
```

**Bước 4:** Viết finally

```
public static void main(String[] args) {  
    try {  
        Scanner nhap = new Scanner(System.in);  
        int soA = nhap.nextInt();  
        int soB = nhap.nextInt();  
  
        try {  
            System.out.println("Phép chia = " + (float)soA / soB);  
        } catch (ArithmeticException ae) {  
            System.out.println("Lỗi tính toán - " + ae.toString());  
        }  
    } catch (InputMismatchException ime) {  
        System.out.println("Dữ liệu nhập không hợp lệ - " + ime.toString());  
    } finally {  
        System.out.println("Dữ liệu nhập an toàn!");  
    }  
}
```

**Bước 5:** Chạy thử chương trình và thử các tình huống gây lỗi.



**Bài thực hành 4:** Viết một chương trình Java yêu cầu người dùng nhập một chuỗi (String) từ bàn phím, nếu không nhập gì thì gán dữ liệu là null, dữ liệu nhập sẽ được ép kiểu về kiểu float, nếu lỗi ép kiểu phải thông báo cùng trong cặp catch với lỗi Null (NullPointerException | NumberFormatException). Lấy số 10 chia cho số vừa nhập chia, bắt lỗi tính toán ở bước này. Kết thúc bắt lỗi Exception chung hiển thị "Không thể thực hiện thao tác, vui lòng báo quản trị viên.", finally hiển thị thông báo "Chương trình kết thúc."

**Chú giải:** Mục tiêu bài thực hành để sinh viên nắm vững kỹ thuật nhóm những lỗi có hành động xử lý tương đồng vào một nhóm, tiết kiệm và tối ưu hóa mã nguồn.

**Bước 1:** Viết mã yêu cầu người dùng nhập dữ liệu từ bàn phím kiểu String.

```
package demo.jp1.lab10.baithuchanh;

import java.util.Scanner;

/**
 *
 * @authorminhvuvc
 */
public class BaiThucHanh04 {

    /**
     * @param args thecommandlinearguments
     */
    public static void main(String[] args) {
        String number = null;
        Scanner nhap = new Scanner(System.in);
        String temp = null;
        if ((temp = nhap.nextLine()).isEmpty()) {
```



```
        number = null;
    } else {
        number = temp;
    }
}
}
```

Bước 2: Viết mã ép kiểu dữ liệu và bắt lỗi kép (2 lỗi trong cùng 1 catch).

```
public static void main(String[] args) {
    try {
        String number = null;
        Scanner nhap = new Scanner(System.in);
        String temp = null;
        if ((temp = nhap.nextLine()).isEmpty()) {
            number = null;
        } else {
            number = temp;
        }
        System.out.println("temp = [" + temp + "]");
        float soThuc = Float.parseFloat(number);
        System.out.println("Số thực: " + soThuc);
    } catch (NullPointerException | NumberFormatException e) {
        System.out.println("Dữ liệu không hợp lệ");
    }
}
```

**Bước 3:** Thực hiện lấy 10 chia và bắt lỗi tính toán.

```
        System.out.println("temp = [" + temp + "]");
        float soThuc = Float.parseFloat(number);
        System.out.println("Số thực: " + soThuc);

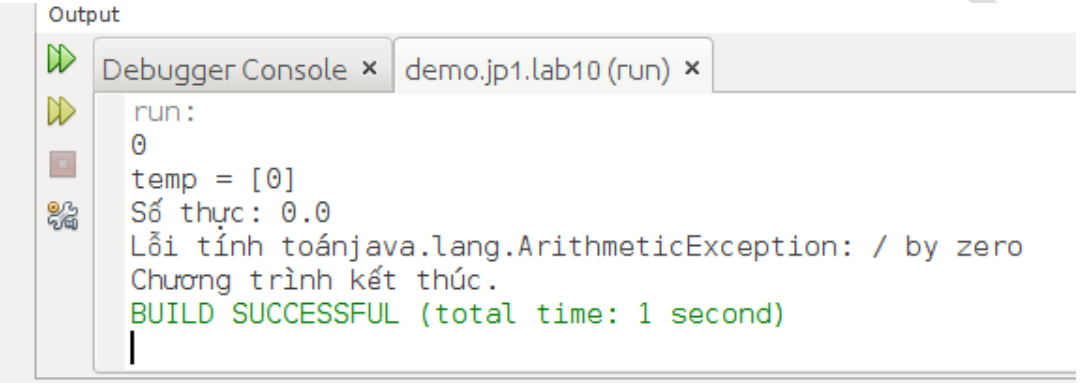
        System.out.println("Phép chia = " + (10 / (int) soThuc));
    } catch (NullPointerException | NumberFormatException e) {
        System.out.println("Dữ liệu không hợp lệ");
    } catch (ArithmeticException arExp) {
        System.out.println("Lỗi tính toán");
    }
}
```

**Bước 4:** Đặt catch Eception chung và finally thông báo.

```
    } catch (NullPointerException | NumberFormatException e) {
        System.out.println("Dữ liệu không hợp lệ");
    }
```

```
} catch (ArithmeticException arExp) {  
    System.out.println("Lỗi tính toán");  
} catch (Exception e) {  
    System.out.println("Không thể thực hiện thao tác, vui lòng báo quản  
trị viên.");  
} finally {  
    System.out.println("Chương trình kết thúc.");  
}
```

**Bước 5:** Chạy chương trình và thử các tính huống lỗi.



```
Output  
Debugger Console x demo.jp1.lab10 (run) x  
run:  
0  
temp = [0]  
Số thực: 0.0  
Lỗi tính toán  
java.lang.ArithmeticException: / by zero  
Chương trình kết thúc.  
BUILD SUCCESSFUL (total time: 1 second)
```

**Chú giải:** *finally cần để xử lý khi mà lỗi có hoặc không xảy ra.*

## Phần II - Bài tập tự làm

**Bài 1:** Làm bài tập theo yêu cầu như sau.

1. Tạo một interface Icar có các phương thức sau:
  - Phương thức "calculateTax()", trả về kiểu float: Tính toán thuế của xe.
  - Phương thức "calculatePrice()", trả về kiểu float: Tính toán tổng giá bán của xe.
  - Phương thức "getInfor()", kiểu trả về void: Hiển thị thông tin của xe.
2. Tạo một lớp Car kế thừa giao diện Icar và bổ sung vào các thuộc tính:
  - private String name;
  - private String producer;
  - private int year;

- private int seat;
- private float rootPrice;

Thực thi phương thức calculateTax() để tính thuế theo công thức sau:

Nếu xe có dưới 7 chỗ ngồi thì thuế = RootPrice \* 60%.

Ngược lại thuế = RootPrice \* 70%

Thực thi phương thức calculatePrice() để tính giá bán xe như sau:

Tổng giá = RootPrice + Tax

Thực thi phương thức getInfor() để hiển thị thông tin theo mẫu:

Xe .... được sản xuất bởi ... vào năm ... có ... ghế với tổng giá là ....\$. Các phần ... tương ứng với các thông số ở trên (Ví dụ: Xe Ford được sản xuất bởi Ford vào năm 1997 có 4 ghế với tổng giá là 20000 \$).

3. Tạo lớp LuxuryCar kế thừa từ lớp Car, bổ sung thêm thuộc tính:

private float specialRate;

Cài đặt phương thức calculatePrice để tính tổng giá bán như sau:

Tổng giá = RootPrice + thuế + RootPrice \* specialRate

Nạp chồng phương thức calculatePrice với 1 tham số "transportCost" có kiểu float.

Tính lại tổng giá bán như sau:

Tổng giá = RootPrice + thuế + RootPrice \* specialRate + transportCost.

Tạo lớp Test.

Tạo menu sau và cài đặt các chức năng của menu:

1. Nhập vào một danh sách LuxuryCar.
2. Hiển thị thông tin của danh sách
3. Sắp xếp danh sách giảm dần bởi giá bán và hiển thị thông tin
4. Tìm kiếm thông tin theo tên xe nhập vào
5. Hiển thị tổng giá bán với giá vận chuyển là \$ 20,000
6. Kết thúc