Exercise 1: Static

Question 1: static variable

Khai báo 1 class student có các thuộc tính id, name, college

Với college là static variable.

Hãy khởi tạo các student sau:

Student có id = 1, name ="Nguyễn Văn A"

Student có id = 2, name = " Nguyễn Văn B "

Student có id = 3, name = " Nguyễn Văn C "

Và tất cả các student này đều học ở "Đại học bách khoa".

Dùng vòng for để in ra thông tin các student

Sau đó hãy chuyển các student này sang "Đại học công nghệ"

Dùng vòng for để in ra thông tin các student

Tạo class Student

```
package com.vti.entity;

import com.vti.ultis.ScannerUltis;

public class Student {
	private int id;
	private String name;
	public static String colllect = "Đại học Bách Khoa";
	private static int COUNT = 0;

	public Student() {
		super();
		this.id = ++COUNT;
		System.out.println("Nhập vào tên sinh viên: ");
		this.name = ScannerUltis.inputString();
	}

	@Override
	public String toString() {
		return "Student [id=" + id + ", name=" + name + " collect:" +
colllect +"]";
	}

}
```

Tạo Class question1

```
package com.vti.backend;

import com.vti.entity.Student;

public class Exercise1 {

	public void question1() {
		Student[] stusentArray = new Student[3];
		System.out.println("khởi tạo 3 sinh viên");
		for (int i = 0; i < 3; i++) {
			System.out.println("Sinh viên " + (i + 1 + ":"));
			Student st = new Student();
			stusentArray[i] = st;
		}
```

```
		System.out.println("Thông tin các sinh viên vừa nhập: ");
		for (int i = 0; i < stusentArray.length; i++) {
			System.out.println(stusentArray[i]);
		}

		System.out.println("Chuyển các sinh viên sang Đại học công nghệ:
");
		Student.colllect = "Đại học Công nghệ ";
		System.out.println("Thông tin sinh viên sau khi chuyển ");
		for (int i = 0; i < stusentArray.length; i++) {
			System.out.println(stusentArray[i]);
		}

	}

}
```

Tạo Class Demo_Exercise1

```
package com.vti.frontend;

import com.vti.backend.Exercise1;

public class Demo_Exercise1 {
	public static void main(String[] args) {
		Exercise1 ex1 = new Exercise1();
		ex1.question1();
	}
}
```

Question 2: tiếp tục question 1

Bổ sung thuộc tính moneyGroup cho Student (moneyGroup là tiền quỹ lớp - dùng chung cho tất cả các student).

Hãy viết chương trình main() để mô tả các bước sau:

B1: Các Student sẽ nộp quỹ, mỗi Student 100k

B2: Student thứ 1 lấy 50k đi mua bim bim, kẹo về liên hoan

B3: Student thứ 2 lấy 20k đi mua bánh mì

B4: Student thứ 3 lấy 150k đi mua đồ dùng học tập cho nhóm

B5: cả nhóm mỗi người lại đóng quỹ mỗi người 50k

In ra số tiền còn của nhóm tại mỗi bước

```
public void question2() {
		Student[] stusentArray = new Student[3];
		System.out.println("khởi tạo 3 sinh viện");
		for (int i = 0; i < 3; i++) {
			System.out.println("Sinh viên " + (i + 1 + ":"));
			Student st = new Student();
			stusentArray[i] = st;
		}
		System.out.println("Các sinh viên nộp quỹ, mỗi bạn 100K: ");
		System.out.println("Tổng quỹ: " + (Student.moneyGroup += 300));

		System.out.println("Student thứ 1 lấy 50k đi mua bim bim, kẹo về
liên hoan: ");
		System.out.println("Tổng quỹ: " + (Student.moneyGroup -= 50));

		System.out.println("Student thứ 2 lấy 20k đi mua bánh mì: ");
```

```
            System.out.println("Tổng quỹ: " + (Student.moneyGroup -= 20));

            System.out.println("Student thứ 3 lấy 150k đi mua đồ dùng học tập
cho nhóm: ");
            System.out.println("Tổng quỹ: " + (Student.moneyGroup -= 150));

            System.out.println("Cả nhóm mỗi người lại đóng quỹ mỗi người 50k:
");
            System.out.println("Tổng quỹ: " + (Student.moneyGroup += 150));
      }
```

Question 3: static method

```
4⊖      public static void main(String[] args) {
5           int result = Math.max(5, 7);
6           System.out.println(result);
7       }
8   }
```

```
Problems  Javadoc  Declaration  Search  Console
<terminated> Program2 [Java Application] C:\Program Files\Java
7
```

a) Viết class MyMath để thay thế cho class Math của java.
b) Viết thêm method min(), sum vào class MyMath
➔Tạo MyMath

```
package com.vti.entity;

public class MyMath {

      public static int max(int a, int b) {
            if (a <= b) {
                  return b;
            } else {
                  return a;
            }
      }

      public static int min(int a, int b) {
            if (a <= b) {
                  return a;
            } else {
                  return b;
            }
      }

      public static int sum(int a, int b) {
            return a + b;

      }
```

```
}
```

➔ question3

```
public void question3() {
		System.out.println("Nhập vào số int 1:");
		int a = ScannerUltis.inputInt();
		System.out.println("Nhập vào số int 2:");
		int b = ScannerUltis.inputInt();

		System.out.println("Max a và b: "+ MyMath.max(a, b));
		System.out.println("Min a và b: "+ MyMath.min(a, b));
		System.out.println("Sum a và b: " + MyMath.sum(a, b));
	}
```

Question 4: tiếp tục Question 1
Trong class Student
a) Viết method cho phép thay đổi college
b) Viết method cho phép lấy giá trị của college

```
	public String getCollect() {
		return Student.colllect;
	}

	public void SetCollect(String newCollect) {
		Student.colllect = newCollect;
	}
```

```
	public void question4() {
		System.out.println("Demo Get Set Collect: ");
		Student st = new Student();
		System.out.println("Collect là: " + st.getCollect());
		System.out.println("Thay đổi Collect: Đại học Hà Nội");
		st.SetCollect("Đại học Hà Nội");
		System.out.println("Collect là: " + st.getCollect());

	}
```

Question 5:
Hãy viết chương trình đếm số Student được sinh ra (tham khảo code trên google)

```
public class Student {
    private int id;
    private String name;
    public static String colllect = "Đại học Bách Khoa";
    public static int COUNT = 0;
    public static int moneyGroup = 0;
```

```
public void question5() {
		System.out.println("Sử dụng For để tạo sinh viên: ");
		for (int i = 0; i < 3; i++) {
			System.out.println("Sinh viên " + (i + 1) + ":");
			Student st = new Student();
		}
```

```
            System.out.println("Sử dụng hàm khởi tạo để tạo sinh viên");
            Student st2 = new Student();
            Student st3 = new Student();
            System.out.println("Số sinh viên được tạo ra trên hệ thống là: " +
Student.COUNT);
      }
```

Question 6: tiếp tục Question 5
Tạo class PrimaryStudent, SecondaryStudent, hãy viết chương trình đếm số lượng PrimaryStudent được sinh ra, SecondaryStudent được sinh ra.
Viết chương trình demo.
Khởi tạo 6 Student, trong đó có 2 PrimaryStudent và 4 SecondaryStudent, sau đó in ra số lượng Student, PrimaryStudent, SecondaryStudent được sinh ra.
➔Tạo Class PrimaryStudent

```
package com.vti.entity;

public class PrimaryStudent extends Student {
      public static int COUNTPri = 0;

      public PrimaryStudent() {
            super();
            COUNTPri++;
      }

}
```

➔Tạo Class SecondaryStudent

```
package com.vti.entity;

public class SecondaryStudent extends Student {
      public static int COUNTSecond = 0;

      public SecondaryStudent() {
            super();
            COUNTSecond++;
      }

}
```

```
public void question6() {
            System.out.println("Tạo 2 Primary Student: ");
            PrimaryStudent pSt1 = new PrimaryStudent();
            PrimaryStudent pSt2 = new PrimaryStudent();
            System.out.println("Tạo 6 Secondary Student: ");
            SecondaryStudent sST1 = new SecondaryStudent();
            SecondaryStudent sST2 = new SecondaryStudent();
            SecondaryStudent sST3 = new SecondaryStudent();
            SecondaryStudent sST4 = new SecondaryStudent();
            SecondaryStudent sST5 = new SecondaryStudent();
            SecondaryStudent sST6 = new SecondaryStudent();
```

```
            System.out.println("Thông tin số lượng sinh viên");
            String leftAlignFormat = "| %-18s | %-4d |%n";

            System.out.format("+--------------------+------+%n");
            System.out.format("| Category           | SL   |%n");
            System.out.format("+--------------------+------+%n");

            System.out.format(leftAlignFormat, "Student", Student.COUNT);
            System.out.format(leftAlignFormat, "PrimaryStudent",
PrimaryStudent.COUNTPri);
            System.out.format(leftAlignFormat, "SecondaryStudent",
SecondaryStudent.COUNTSecond);

            System.out.format("+--------------------+------+%n");
      }
```

Question 7: tiếp tục Question 6
Chỉ cho phép user tạo được tối đa 7 học sinh

```
public Student() {
            super();
            COUNT++;
            if (COUNT > 7) {
                  System.err.println("Số lượng sinh viên đã vượt quá 7, không
thể tạo thêm sinh viên");
            } else {
                  this.id = COUNT;
                  System.out.println("Nhập vào tên sinh viên: ");
                  this.name = ScannerUltis.inputString();
            }

      }
```

```
public void question7() {
            for (int i = 0; i < 6; i++) {
                  Student st = new Student();
            }
            System.out.println("Tạo Primary Student: ");
            PrimaryStudent pSt = new PrimaryStudent();
            System.out.println("Sinh viên Primary Student: " + pSt);

            System.out.println("Tạo Secondary Student: ");
            SecondaryStudent sSt = new SecondaryStudent();
            System.out.println("Sinh viên Secondary Student: " + sSt);
      }
```

Question 8:
Tạo class Configs có các static property
SO_LUONG_HINH_TOI_DA = 5;
Tạo class HinhHoc, và class HinhTron, HinhChuNhat sẽ kế thừa từ class HinhHoc, implement method tính chu vi, diện tính
Viết chương trình chỉ cho phép khởi tạo được tối đa 5 hình, nếu người dùng khởi tạo tới hình thứ 6 thì sẽ throw ra 1 custom Exception có tên là HinhHocException có message là: "Số lượng hình tối đa là: " + Configs. SO_LUONG_HINH_TOI_DA

➔Tạo Class Config

```
package com.vti.entity;

public class Config {
	public static int SO_LUONG_HINH_TOI_DA = 5;
	public static Float PI = (float) 3.14;
}
```

➔ Tạo Class HinhHocException

```
package com.vti.entity;

public class HinhHocException extends Exception {

	public HinhHocException() {
		super("Số lượng hình đã tạo vượt quá số lượng cho phép là: " +
Config.SO_LUONG_HINH_TOI_DA
				+ " hãy kiểm tra lại.");
	}

}
```

➔Tạo Class HinhHoc:

```
package com.vti.entity;

public abstract class HinhHoc {
	private Float a;
	private Float b;
	public static int COUNT = 0;

	public abstract Float tinhChuVI(Float a, Float b);

	public abstract Float tinhDienTich(Float a, Float b);

	public HinhHoc(Float a, Float b) throws Exception {
		super();
		COUNT++;
		if (COUNT <= Config.SO_LUONG_HINH_TOI_DA) {
			this.a = a;
			this.b = b;
		} else {
			throw new HinhHocException();
		}
```

```
	}

}
```

➔Tạo Class HinhTron

```
package com.vti.entity;

public class HinhTron extends HinhHoc {

	public HinhTron(Float a, Float b) throws Exception {
		super(a, b);
		// TODO Auto-generated constructor stub
	}

	@Override
	public Float tinhChuVI(Float a, Float b) {

		return 2*a*(Config.PI);
	}

	@Override
	public Float tinhDienTich(Float a, Float b) {

		return (float) (Config.PI*(Math.pow(a, 2)));
	}

}
```

➔Tạo Class HinhChuNhat

```
package com.vti.entity;

public class HinhChuNhat extends HinhHoc {

	public HinhChuNhat(Float a, Float b) throws Exception {
		super(a, b);
	}

	@Override
	public Float tinhChuVI(Float a, Float b) {

		return 2 * (a + b);
	}

	@Override
	public Float tinhDienTich(Float a, Float b) {

		return (a * b);
	}

}
```

```
public void question8() throws Exception {
		System.out.println("Tạo 4 hình chữ nhật.");
		HinhChuNhat[] hcns = new HinhChuNhat[4];
		for (int i = 0; i < 4; i++) {
```

```
                System.out.println("Hình " + (i + 1) + ":");
                System.out.println("Canh a: ");
                Float a = ScannerUltis.inputFloat();
                System.out.println("Canh b: ");
                Float b = ScannerUltis.inputFloat();
                HinhChuNhat hcn = new HinhChuNhat(a, b);
                hcns[i] = hcn;
            }
            System.out.println("Tạo hình tròn số 1, nhập vào bán kính ");
            Float r1 = ScannerUltis.inputFloat();
            HinhTron hinhtron1 = new HinhTron(r1, r1);

            System.out.println("Tạo hình tròn số 2, nhập vào bán kính ");
            Float r2 = ScannerUltis.inputFloat();
            HinhTron hinhtron2 = new HinhTron(r2, r2);
        }
```

Question 9:

Thay đổi các method ở class ScannerUtils ở Assignment 6 thành static để tiện dùng hơn

```
package com.vti.ultis;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.util.Scanner;
import java.util.regex.Pattern;

public class ScannerUltis {
	private static Scanner sc = new Scanner(System.in);

	public static int inputInt() {
		while (true) {
			try {
				return Integer.parseInt(sc.next().trim());
			} catch (Exception e) {
				System.err.println("Nhập lại:");
			}
		}
	}

	public static int inputIntPositive() {
	    while (true) {
			try {
				int intPositive = Integer.parseInt(sc.next());
				if (intPositive >= 0) {
					return intPositive;
				} else {
					System.err.println("Nhập lại:");
				}

			} catch (Exception e) {
				System.err.println("Nhập lại:");
			}

		}
	}
```

```
	public static Float inputFloat(String mes) {
		while (true) {
			try {
				return Float.parseFloat(sc.next());
			} catch (Exception e) {
				System.err.println("Nhập lại:");
			}
		}
	}

	public static Double inputDouble() {
		while (true) {
			try {
				return Double.parseDouble(sc.next());
			} catch (Exception e) {
				System.err.println("Nhập lại:");
			}
		}
	}

	public static String inputString() {
		while (true) {
			String string = sc.nextLine().trim();
			if (!string.isEmpty()) {
				return string;
			} else {
				System.err.println("Nhập lại:");
			}
		}
	}

	public static LocalDate inputLocalDate() {
		System.out.println("Nhập theo định dạng yyyy-MM-dd");
		SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
		while (true) {
			String localdate = sc.next().trim();
			try {
				if (format.parse(localdate) != null) {
					LocalDate lc = LocalDate.parse(localdate);
					return lc;
				}
			} catch (Exception e) {
				System.err.println("Nhập lại:");
			}

		}
	}
}
```


Exercise 2 (Optional): Final

Question 1: final variable

Tạo class MyMath, khai báo final variable số PI = 3.14

Viết method sum(int a) và trả về tổng của a và PI

Thử thay đổi số PI = 3.15 trong method xem có được ko?

➔Tạo Class MyMath

```java
package com.vti.entity;

public class MyMath {
	public static final Double PI = 3.14;

	public static double sum(int a) {

		return a + PI;
	}

	public static int max(int a, int b) {
		if (a <= b) {
			return b;
		} else {
			return a;
		}
	}

	public static int min(int a, int b) {
		if (a <= b) {
			return a;
		} else {
			return b;
		}
	}

	public static int sum(int a, int b) {
		return a + b;

	}
}
```

```java
	public void question1() {
		System.out.println("Chương trình tính tổng số int với PI");
		System.out.println("Nhập vào 1 số int: ");
		int a = ScannerUltis.inputInt();
		System.out.println("Tổng với PI là: " + MyMath.sum(a));
//		System.out.println("Thực hiện thay đổi số PI = 3.15");  ==> Eclip sẽ báo lỗi
//		MyMath.PI = 3.15;

	}
```


Question 2:

Tạo class Student có property id, name hãy thiết kế class Student sao cho khi đã khởi tạo Student thì id là không thể thay đổi trong suốt quá trình chạy chương trình chương trình

➔Tạo Class StudentEx2 ➔ đặt final cho ID thì trong class này sẽ không thể tạo phương thức setter cho id;

```java
package com.vti.entity;

public class StudentEx2 {
	private final int id;
	private String name;
```

```
	public StudentEx2(int id, String name) {
		super();
		this.id = id;
		this.name = name;
	}

	/**
	 * @return the name
	 */
	public String getName() {
		return name;
	}

	/**
	 * @param name the name to set
	 */
	public void setName(String name) {
		this.name = name;
	}

	/**
	 * @return the id
	 */
	public int getId() {
		return id;
	}

	@Override
	public String toString() {
		return "StudentEx2 [id=" + id + ", name=" + name + "]";
	}

}
```

```
	public void question2() {
		System.out.println("Khởi tạo Student");
		System.out.println("Nhập vào ID:");
		int id = ScannerUltis.inputInt();
		System.out.println("Nhập vào Name:");
		String name = ScannerUltis.inputString();
		StudentEx2 stEx2 = new StudentEx2(id, name);
		System.out.println("Thông tin sinh viên vừa nhập: ");
		System.out.println(stEx2);
	}
```

Question 3: Tiếp tục Question 2 (final method)
Student bắt buộc phải học bài, và việc học bài là như nhau đối với các học sinh ta viết method void study(), bên trong method ta sẽ in ra text "Đang học bài…"
Tiếp theo ta có các loại Student như sau: PrimaryStudent, SecondaryStudent, và có thể còn có nhiều loại học sinh khác trong tương lai, tất cả các student này đều kế thừa Student
Hãy thiết kế chương trình để tất cả các học sinh đều có method study() như nhau (nghĩa là PrimaryStudent, SecondaryStudent không thể override được method study())

```java
package com.vti.entity;

public class StudentEx2 {
	private final int id;
	private String name;

	public StudentEx2(int id, String name) {
		super();
		this.id = id;
		this.name = name;
	}

	public static void study() {
		System.out.println("Sinh viên đang học bài");
	}

	/**
	 * @return the name
	 */
	public String getName() {
		return name;
	}

	/**
	 * @param name the name to set
	 */
	public void setName(String name) {
		this.name = name;
	}

	/**
	 * @return the id
	 */
	public int getId() {
		return id;
	}

	@Override
	public String toString() {
		return "StudentEx2 [id=" + id + ", name=" + name + "]";
	}

}
```

➔ Chương trình demo này chưa chạy được ➔ tìm nguyên nhân, chưa gọi được Study() từ PrimaryStusent và SecondaryStudent;

```
	public void question3() {
		PrimaryStudent priStudent = new PrimaryStudent();
		SecondaryStudent secondStudent = new SecondaryStudent();
//		System.out.println("priStudent: "+ PrimaryStudent.s);

	}
```

Question 4: Tiếp tục Question 3 (final class)
Hãy thiết kế chương trình sao cho không có class nào có thể kế thừa từ PrimaryStudent, SecondaryStudent

```
package com.vti.entity;

public final class PrimaryStudentEx2 extends Student {

	public PrimaryStudentEx2() {
		super();
		// TODO Auto-generated constructor stub
	}

}
```

```
package com.vti.entity;

public final class SecondaryStudentEx2 extends Student {

	public SecondaryStudentEx2() {
		super();
		// TODO Auto-generated constructor stub
	}

}
```


Exercise 3 (Optional): File

Tạo class FileManager ở package utils, sau đó tất cả các Question trong Exercise này sẽ viết vào trong class FileManager, với mỗi Question sẽ viết demo ở front-end
Chú ý: tất cả các error message nên để là constant (static final String)
Tạo class FileUltis trong Package ultis:

```
package com.vti.ultis;

import java.io.File;
import java.nio.file.Files;
import java.util.Arrays;
import java.util.List;

public class FileUltis {
	public static final String FILE_EXISTS = "File is exits!";
	public static final String FILE_NOT_EXISTS = "Error! File Not Exist.";
```

```
	public static final String FOLDER_EXISTS = "Folder is exits!";
	public static final String FOLDER_NOT_EXISTS = "Folder is not exits!";
	public static final String PATH_NOT_FOLDER = "Error! Path is not folder.";

	public static final String SOURCE_FILE_NOT_EXISTS = "Source File is not exits!";
	public static final String DESTINATION_FILE_EXISTS = "Destination File is exits!";

	public static final String NEW_FILE_EXISTS = "Error! New File Exist.";
	public static final String CREATE_FILE_SUCCESS = "Create file success!";
	public static final String CREATE_FILE_FAIL = "Create file fail!";
	public static final String DELETE_FILE_SUCCESS = "Delete file success!";
	public static final String DELETE_FILE_FAIL = "Delete file fail!";

	public static boolean isFileExists(String pathFile) {
		return new File(pathFile).exists() ? true : false;
	}

	// Check folder is exists
	public static boolean isFolderExists(String pathFolder) {
		return new File(pathFolder).exists() ? true : false;
	}

	// Create file
	public static void createNewFile(String pathFile) throws Exception {
		if (isFileExists(pathFile)) {
			throw new Exception(FILE_EXISTS);
		}

		boolean result = new File(pathFile).createNewFile();

		System.out.println(result ? CREATE_FILE_SUCCESS : CREATE_FILE_FAIL);
	}

	public static void createNewFile(String path, String fileName) throws Exception {
		String pathFile = path + "//" + fileName;
		createNewFile(pathFile);
	}

	// Delete file
	public static void deleteFile(String pathFile) throws Exception {
		if (!isFileExists(pathFile)) {
			throw new Exception(FILE_NOT_EXISTS);
		}

		boolean result = new File(pathFile).delete();

		System.out.println(result ? DELETE_FILE_SUCCESS : DELETE_FILE_FAIL);
	}

	// Check path is File or Folder
	public static void isFolderOrFile(String pathFile) {
		File file = new File(pathFile);
		if (file.isDirectory()) {
			System.out.println("Đây là 1 Foder");
		} else if (file.isFile()) {
			System.out.println("Đây là 1 file");
		} else {
			System.out.println("Đây không phải đường dẫn, cũng không phải file");
		}
	}

	// Check path is Folder
	public static boolean isFolder(String pathFile) {
		File file1 = new File(pathFile);
		return file1.isDirectory();
	}

	// Get all file name of folder
	public static String[] getAllFileName(String path) {

		File file = new File(path);
		if (!isFolder(path)) {
			System.out.println("Đây không phải là đường dẫn!!");
			return null;
```

```
                } else {
                        return file.list();
                }

        }

        // Copy File
        public static void copyFile(String sourceFile, String destinationPath) throws Exception {

                if (!isFileExists(sourceFile)) {
                        throw new Exception(SOURCE_FILE_NOT_EXISTS);
                }

                String[] s = sourceFile.split("/");
                String nameFile = s[s.length - 1];

                String destinationFile = destinationPath + "//" + nameFile;

                if (isFileExists(destinationFile)) {
                        throw new Exception(DESTINATION_FILE_EXISTS);
                }

                File source = new File(sourceFile);
                File dest = new File(destinationPath);

                Files.copy(source.toPath(), dest.toPath());
        }

        // Moving file
        public static void moveFile(String sourceFile, String destinationPath) throws Exception {

                if (!isFileExists(sourceFile)) {
                        throw new Exception(SOURCE_FILE_NOT_EXISTS);
                }

                File source = new File(sourceFile);
                File dest = new File(destinationPath);
                Files.move(source.toPath(), dest.toPath());
        }

        // Rename File
        public static void renameFile(String pathFile, String newName) throws Exception {

                if (!isFileExists(pathFile)) {
                        throw new Exception(FILE_NOT_EXISTS);
                }

                if (isFileExists(newName)) {
                        throw new Exception(NEW_FILE_EXISTS);
                }
                File oldFile = new File(pathFile);
                File newFile = new File(newName);
                oldFile.renameTo(newFile);
        }

        // Create New Folder
                public static void createNewFolder(String newPathFolder) throws Exception {
                        if (isFolderExists(newPathFolder)) {
                                throw new Exception(FOLDER_EXISTS);
                        }
                        new File(newPathFolder).mkdir();
                }
}
```


Tạo class ScannerUltis trong Package ultis:

```
package com.vti.ultis;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.util.Scanner;
```

```java
import java.util.regex.Pattern;

public class ScannerUltis {
	private static Scanner sc = new Scanner(System.in);

	public static int inputInt() {
		while (true) {
			try {
				return Integer.parseInt(sc.next().trim());
			} catch (Exception e) {
				System.err.println("Nhập lại:");
			}
		}
	}

	public static int inputIntPositive() {
		while (true) {
			try {
				int intPositive = Integer.parseInt(sc.next());
				if (intPositive >= 0) {
					return intPositive;
				} else {
					System.err.println("Nhập lại:");
				}

			} catch (Exception e) {
				System.err.println("Nhập lại:");
			}

		}
	}

	public static Float inputFloat() {
		while (true) {
			try {
				return Float.parseFloat(sc.next());
			} catch (Exception e) {
				System.err.println("Nhập lại:");
			}
		}
	}

	public static Double inputDouble() {
		while (true) {
			try {
				return Double.parseDouble(sc.next());
			} catch (Exception e) {
				System.err.println("Nhập lại:");
			}
		}
	}

	public static String inputString() {
		while (true) {
			String string = sc.next().trim();
			if (!string.isEmpty()) {
				return string;
			} else {
				System.err.println("Nhập lại:");
			}
		}
	}

	public static LocalDate inputLocalDate() {
		System.out.println("Nhập theo định dạng yyyy-MM-dd");
		SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
		while (true) {
			String localdate = sc.next().trim();
			try {
				if (format.parse(localdate) != null) {
					LocalDate lc = LocalDate.parse(localdate);
					return lc;
				}
			} catch (Exception e) {
```

```
                                System.err.println("Nhập lại:");
                        }

                }
        }
}
```

Tạo class demo:

```
import com.vti.ultis.FileUltis;
import com.vti.ultis.ScannerUltis;

public void question1() throws Exception {
                loadMenuFile();
        }

        private void loadMenuFile() throws Exception {
                while (true) {

        System.out.println("================================================================================
");
                        System.out.println("=================Lựa chọn chức năng bạn muốn sử
dụng==================");
                        System.out.println("===              1. Check File is exists.
===");
                        System.out.println("===              2. Check Folder
===");
                        System.out.println("===              3. Tạo file mới
===");
                        System.out.println("===              4. Tạo mới file từ đường dẫn và tên
file riêng     ===");
                        System.out.println("===              5. Xóa file
===");
                        System.out.println("===              6. Check path is File or Folder
===");
                        System.out.println("===              7. Get all file in Folder
===");
                        System.out.println("===              8. Copy File
===");
                        System.out.println("===              9. Move File
===");
                        System.out.println("===             10. Rename File
===");
                        System.out.println("===             11. Tạo Folder
===");
                        System.out.println("===             12. Download File
===");
                        System.out.println("===             15. End Program
===");

        System.out.println("================================================================================
");
                        int menuChoose = ScannerUltis.inputIntPositive();
                        switch (menuChoose) {
                        case 1:
                                System.out.println("Nhập vào đường dẫn và file cần kiểm tra: ");
                                String path1 = ScannerUltis.inputString();
                                if (FileUltis.isFileExists(path1)) {
                                        System.out.println("Có file trên hệ thống");
                                } else {
                                        System.out.println("Không có file trên hệ thống");

                                }
                                ;
                                break;

                        case 2:
                                System.out.println("Nhập vào Folder cần kiểm tra: ");
                                String path2 = ScannerUltis.inputString();
                                if (FileUltis.isFolderExists(path2)) {
                                        System.out.println("Có Folder trên hệ thống");
                                } else {
```

```
                                        System.out.println("Không có Folder trên hệ thống");
                                }
                                break;
                        case 3:
                                System.out.println("Nhập vào đường dẫn và tên file cần tạo: ");
                                String path3 = ScannerUltis.inputString();
                                FileUltis.createNewFile(path3);
                                break;
                        case 4:
                                System.out.println("Nhập vào đường dẫn: ");
                                String path4 = ScannerUltis.inputString();
                                System.out.println("Nhập vào tên file cần tạo: ");
                                String fileName4 = ScannerUltis.inputString();
                                FileUltis.createNewFile(path4, fileName4);
                                break;
                        case 5:
                                System.out.println("Nhập vào đường dẫn và tên file cần xóa: ");
                                String path5 = ScannerUltis.inputString();
                                FileUltis.deleteFile(path5);
                                break;
                        case 6:
                                System.out.println("Nhập vào Path cần kiểm tra ");
                                String path6 = ScannerUltis.inputString();
                                FileUltis.isFolderOrFile(path6);
                                break;
                        case 7:
                                System.out.println("Nhập vào Path cần kiểm tra ");
                                String path7 = ScannerUltis.inputString();
                                String[] list = FileUltis.getAllFileName(path7);
                                for (int i = 0; i < list.length; i++) {
                                        System.out.println(list[i]);
                                }
                                break;
                        case 8:
                                System.out.println("Nhập vào source nguồn: ");
                                String source8 = ScannerUltis.inputString();
                                System.out.println("Nhập vào Folder đích: ");
                                String path8 = ScannerUltis.inputString();
                                FileUltis.copyFile(source8, path8);
                                break;
                        case 9:
                                System.out.println("Nhập vào file nguồn: ");
                                String source9 = ScannerUltis.inputString();
                                System.out.println("Nhập vào folder đích: ");
                                String destination9 = ScannerUltis.inputString();
                                FileUltis.moveFile(source9, destination9);
//                              chưa chạy được chức năng này cần kiểm tra thêm.
                                break;
                        case 10:
                                System.out.println("-- Chú ý nhập vào đầy đủ cả đường dẫn và tên
file.");
                                System.out.println("Nhập vào file nguồn: ");
                                String source10 = ScannerUltis.inputString();
                                System.out.println("Nhập vào file đích: ");
                                String des10 = ScannerUltis.inputString();
                                FileUltis.renameFile(source10, des10);

                                break;

                        case 11:
                                System.out.println("Nhập vào đường dẫn folder cần tạo:  ");
                                String newPathFolder = ScannerUltis.inputString();
                                FileUltis.createNewFolder(newPathFolder);
                                break;
                        case 12:
                                System.out.println("No Data");
                                break;
                        case 15:
                                return;
                        default:
                                System.out.println("Alarm: Lựa chọn đúng số trên menu");
                                break;
                        }
                }
```

```
	}
```

Question 1: Check File is exists
Tạo 1 method có đầu vào là String path để check xem path đó có tồn tại hay không.
VD: path = "C:\Users\pc\Desktop\Test.txt"
Gợi ý: Tạo method **boolean isFileExists(String pathFile)**
File Ultis

```
public static boolean isFileExists(String pathFile) {
		return new File(pathFile).exists() ? true : false;
	}
```

```
	case 1:
				System.out.println("Nhập vào đường dẫn file cần kiểm
tra: ");
				String path = ScannerUltis.inputString();
				if(FileUltis.isFileExists(path)) {
					System.out.println("Có file trên hệ thống");
				}else {
					System.out.println("Không có file trên hệ
thống");

				};
```

Question 2: Create new file
Tạo 1 method để có thể tạo được file.
Chú ý:
Kiểm tra xem file đã tồn tại hay chưa, nếu file đã tồn tại thì sẽ throw ra Exception "Error! File Exist."
Gợi ý:
Tạo method **void createNewFile(String pathFile)**
Class FileUltis

```
public static void createNewFile(String pathFile) throws Exception {
		if (isFileExists(pathFile)) {
			throw new Exception(FILE_EXISTS);
		}

		boolean result = new File(pathFile).createNewFile();

		System.out.println(result ? CREATE_FILE_SUCCESS :
CREATE_FILE_FAIL);
	}

	public static void createNewFile(String path, String fileName) throws
Exception {
		String pathFile = path + "//" + fileName;
		createNewFile(pathFile);
	}
```

```
case 3:
                        System.out.println("Nhập vào đường dẫn và tên file
cần tạo: ");
                        String path3 = ScannerUltis.inputString();
                        FileUltis.createNewFile(path3);
```

Vào tạo thêm method **void createNewFile(String path, String fileName)**

```
      case 4:
                        System.out.println("Nhập vào đường dẫn: ");
                        String path4 = ScannerUltis.inputString();
                        System.out.println("Nhập vào tên file cần tạo: ");
                        String fileName4 = ScannerUltis.inputString();
                        FileUltis.createNewFile(path4, fileName4);
                        break;
```

Question 4: Delete file
Tạo 1 method để có thể delete file
Chú ý:
Kiểm tra xem file đã tồn tại hay chưa, nếu file chưa tồn tại thì sẽ throw ra Exception "Error! File Not Exist."
Gợi ý: Tạo method **void deleteFile(String pathFile)**
Class FileUltis

```
// Delete file
      public static void deleteFile(String pathFile) throws Exception {
            if (!isFileExists(pathFile)) {
                  throw new Exception(FILE_NOT_EXISTS);
            }

            boolean result = new File(pathFile).delete();

            System.out.println(result ? DELETE_FILE_SUCCESS :
DELETE_FILE_FAIL);
      }
```

```
case 5:
                        System.out.println("Nhập vào đường dẫn và tên file
cần xóa: ");
                        String path5 = ScannerUltis.inputString();
                        FileUltis.deleteFile(path5);
                        break;
```

Question 5: Check path is File or Folder
Tạo 1 method để kiểm tra xem path có thể là File hay Folder.
Gợi ý: Tạo method **void isFolderOrFile (String path)**
Class FileUltis

```
// Check path is File or Folder
      public static void isFolderOrFile(String pathFile) {
            File file = new File(pathFile);
            if (file.isDirectory()) {
                  System.out.println("Đây là 1 Foder");
```

```
            } else if (file.isFile()) {
                  System.out.println("Đây là 1 file");
            } else {
                  System.out.println("Đây không phải đường dẫn, cũng không
phải file");
            }
      }
```

```
      case 6:
                        System.out.println("Nhập vào Path cần kiểm tra ");
                        String path6 = ScannerUltis.inputString();
                        FileUltis.isFolderOrFile(path6);
                        break;
```

Question 6: Get all File name of Folder
Tạo 1 method để lấy ra tất cả các tên file trong 1 Folder.
Chú ý:
Kiểm tra xem path nhập vào có phải là folder hay không, nếu không phải thì sẽ throw ra Exception "Error! Path is not folder."
Gợi ý: Tạo method **List<String> getAllFileName(String path)**
Class FileUltis

```
      // Get all file name of folder
      public static String[] getAllFileName(String path) {

            File file = new File(path);
            if (!isFolder(path)) {
                  System.out.println("Đây không phải là đường dẫn!!");
                  return null;
            } else {
                  return file.list();
            }

      }
```

```
case 7:
                        System.out.println("Nhập vào Path cần kiểm tra ");
                        String path7 = ScannerUltis.inputString();
                        String[] list = FileUltis.getAllFileName(path7);
                        for (int i = 0; i < list.length; i++) {
                              System.out.println(list[i]);
                        }
                        break;
```

Question 7: Copy File
Tạo 1 method để copy file.
Chú ý:
Nếu file không tồn tại thì sẽ throw ra Exception "Error! Source File Not Exist."
Nếu file đích đã tồn tại thì sẽ throw ra Exception "Error! newPath has File same name."
Gợi ý:
Tạo method **void copyFile(String sourceFile, String distinationPath, String newName)**
Và tạo method **void copyFile(String sourceFile, String newPath).** Với File mới sẽ có tên cùng với file cũ

Question 8: Moving file
Tạo 1 method để di chuyển file sang folder khác.
Chú ý:
Nếu file không tồn tại thì sẽ throw ra Exception "Error! File Not Exist.".
Phải xóa file cũ đi
Gợi ý:
Tạo method **void moveFile(String sourceFile, String destinationPath)**

Question 9: Rename File
Tạo 1 method để đổi tên file.
Chú ý:
Nếu file không tồn tại thì sẽ throw ra Exception "Error! File Not Exist.".
Nếu newName đã tồn tại thì sẽ throw ra Exception "Error! Name is Exist.".
Class FileUltis:

```
// Rename File
	public static void renameFile(String pathFile, String newName) throws
Exception {

		if (!isFileExists(pathFile)) {
			throw new Exception(FILE_NOT_EXISTS);
		}

		if (isFileExists(newName)) {
			throw new Exception(NEW_FILE_EXISTS);
		}
		File oldFile = new File(pathFile);
		File newFile = new File(newName);
		oldFile.renameTo(newFile);
	}
```

```
case 10:
                        System.out.println("-- Chú ý nhập vào đầy đủ cả đường
dẫn và tên file.");
                        System.out.println("Nhập vào file nguồn: ");
                        String source10 = ScannerUltis.inputString();

                        System.out.println("Nhập vào file đích: ");
                        String des10 = ScannerUltis.inputString();
                        FileUltis.renameFile(source10, des10);

                        break;
```

Gợi ý:
Tạo method **void renameFile(String pathFile, String newName)**

Question 10: Create new folder
Tạo 1 method để có thể tạo được folder.
Chú ý:
Kiểm tra xem folder đã tồn tại hay chưa, nếu folder đã tồn tại thì sẽ throw ra Exception "Error! Folder Exist."
Gợi ý: Tạo method **void createNewFolder(String newPathFolder)**
Class FileUltis:

```
// Create New Folder
        public static void createNewFolder(String newPathFolder) throws
Exception {
                if (isFolderExists(newPathFolder)) {
                        throw new Exception(FOLDER_EXISTS);
                }
                new File(newPathFolder).mkdir();
        }
```

```
      case 11:
                        System.out.println("-- Chú ý nhập vào đầy đủ cả đường
dẫn và tên file.");
                        System.out.println("Nhập vào đường dẫn folder cần
tạo:  ");
                        String newPathFolder = ScannerUltis.inputString();
                        FileUltis.createNewFolder(newPathFolder);
                        break;
```

Question 11: Download File
Hãy viết 1 method để có thể download được file ở trên mạng
Chú ý:
Nếu folder không tồn tại thì sẽ tạo folder.
Nếu tên file download đã tồn tại trong folder thì sẽ thêm (1), (2) vào sau tên file (giống download file ở window).
Gợi ý:
Tạo method **void downloadFile(String fileLink, String folder)**

```
public boolean downloadFile(String link, String folderSave)
        throws MalformedURLException, IOException {
    // if folder not Exist or not folder
    File folder = new File(folderSave);
    if (!folder.exists() || !folder.isDirectory()) {
        // show message error
        System.out.println(FOLDER_NOT_EXIST_OR_NOT_FOLDER);
        return false;
    }
    // if folder exist
    // get name file
    String s[] = link.split("/");
    String name = s[s.length - 1];
    folderSave = folderSave + "/" + name;
    // create connection to URL
    URL url = new URL(link);
    // open connection
    URLConnection connection = url.openConnection();
    // get size of file
    int size = connection.getContentLength();
    // read file from Internet use InputStream
    InputStream in = connection.getInputStream();
    // save file use FileOutputStream
    FileOutputStream output = new FileOutputStream(folderSave);
    int byteDownloaded = 0;
    byte[] b = new byte[1024];
    // get length of file. If not read then length = -1
    int length = in.read(b);
    while (length != -1) {
        byteDownloaded += length;
        // print % byte downloaded
        System.out.println(byteDownloaded * 100f / size + "%");
        // write content downloaded from position 0 -> length to output
        output.write(b, 0, length);
        length = in.read(b);
    }
    // close
    output.close();
    in.close();
    // show message
    System.out.println(DOWNLOAD_FILE_SUCCESS);
    return true;
}
```

Code tham khảo

Code tham khảo

Exercise 4 (Optional): IO Stream

Tạo class IOManager ở package utils, sau đó tất cả các Question trong Exercise này sẽ viết vào trong class IOManager, với mỗi Question sẽ viết demo ở front-end
Chú ý: tất cả các error message nên để là constant (static final String)

Tạo class IOManager trong Utils:

```
package com.vti.ultis;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
```

```java
public class IOManager {

	private static final String FILE_NOT_EXIST = "Error! File not Exist.";
	private static final String WRITE_FILE_SUCCESS = "Write file success!";
	private static final String READ_FILE_SUCCESS = "Read file success!";

	public static String readFile(String pathFile) throws Exception {
		if (!FileUltis.isFileExists(pathFile)) {
			throw new Exception(FILE_NOT_EXIST);
		}
		byte[] b = new byte[1024];
		FileInputStream in = new FileInputStream(pathFile);
		int length = in.read(b);
		String output = "";
		while (length > -1) {
			String content = new String(b, 0, length);
//			System.out.println(content + "\n " + length);
			output += content;
			length = in.read(b);
		}
		in.close();
		return output;
	}

	public static void writeFile(String pathFile, boolean isContinuing, String content) throws
Exception {
		if (!FileUltis.isFileExists(pathFile)) {
			throw new Exception(FILE_NOT_EXIST);
		}
		FileOutputStream out = new FileOutputStream(pathFile, isContinuing);
		out.write(content.getBytes());
		out.close();
		System.out.println(WRITE_FILE_SUCCESS);
	}

	public static void writeObject(Object object, String path, String fileName) throws
Exception {
		if (!FileUltis.isFileExists(path)) {
			throw new Exception(FILE_NOT_EXIST);
		}
		System.out.print("File name:");
		fileName = ScannerUltis.inputString();
		path = path + "\\" + fileName;
		FileOutputStream out = new FileOutputStream(path);
		ObjectOutputStream objectOut = new ObjectOutputStream(out);
		objectOut.writeObject(object);
		out.close();
		objectOut.close();
		System.out.println(WRITE_FILE_SUCCESS);
	}

	public static void writeObject(Object object, String path) throws Exception {
		if (!FileUltis.isFileExists(path)) {
			throw new Exception(FILE_NOT_EXIST);
		}
		FileOutputStream out = new FileOutputStream(path);
		ObjectOutputStream objectOut = new ObjectOutputStream(out);
		objectOut.writeObject(object);
		out.close();
		objectOut.close();
		System.out.println(WRITE_FILE_SUCCESS);
	}

	public static Object readObject(String filePath) throws Exception {
		if (!FileUltis.isFileExists(filePath)) {
			throw new Exception(FILE_NOT_EXIST);
		}
		FileInputStream in = null;
		ObjectInputStream objectIn = null;
		try {
			in = new FileInputStream(filePath);
			objectIn = new ObjectInputStream(in);
			System.out.println(READ_FILE_SUCCESS);
```

```
                        return objectIn.readObject();
                } finally {
                        in.close();
                        objectIn.close();
                }

        }
}
```

Tạo class Exercise_4 trong Backend:

```
package com.vti.backend;

import com.vti.ultis.FileUltis;
import com.vti.ultis.IOManager;
import com.vti.ultis.ScannerUltis;

public class Exercise_4 {
        public void question_1() throws Exception {
                System.out.println("Chương trình đọc file:");
                System.out.println("Nhập vào file muốn đọc:");
                String path = ScannerUltis.inputString();
                System.out.println(IOManager.readFile(path));
        };

        public void question_2() throws Exception {
                System.out.println("Chương trình ghi file:");
                System.out.println("Nhập vào file muốn ghi:");
                String path = ScannerUltis.inputString();
                System.out.println("Nhập vào nội dung muốn ghi:");
                String content = ScannerUltis.inputString();
                IOManager.writeFile(path, true, content);
        }
}
```

Tạo class Demo_Exercise_4 trong Frontend:

```
package com.vti.frontend;

import com.vti.backend.Exercise_4;

public class Demo_Exercise_4 {
        public static void main(String[] args) throws Exception {
                Exercise_4 ex4 = new Exercise_4();
//              ex4.question_1();
                ex4.question_2();
        }
}
```

Question 1: Tạo method read File
Tạo 1 method có đầu vào là String filePath và return ra nội dung trong file.
Chú ý:
Nếu file không tồn tại thì sẽ throw ra Exception "Error! File Not Exist." (sử dụng FileManager ở trên để check)
Gợi ý: Tạo method **String readFile(String pathFile)**

```
	public static String readFile(String pathFile) throws Exception {
		if (!FileUltis.isFileExists(pathFile)) {
			throw new Exception(FILE_NOT_EXIST);
		}
		byte[] b = new byte[1024];
		FileInputStream in = new FileInputStream(pathFile);
		int length = in.read(b);
		String output = "";
		while (length > -1) {
			String content = new String(b, 0, length);
//			System.out.println(content + "\n " + length);
			output += content;
			length = in.read(b);
		}
		in.close();
		return output;
	}
```

Question 2: Tạo method write content to File
Tạo 1 method để write file
Chú ý:
Nếu file không tồn tại thì sẽ throw ra Exception "Error! File Not Exist." (sử dụng FileManager ở exercise trên để check)
Gợi ý: Tạo method **void writeFile(String pathFile, isContinuing String content)**
Nếu isContinuing = false thì sẽ xóa hết nội dung trong file đi và write content mới vào
Nếu isContinuing = true thì sẽ ghi content vào cuối file

```
public static void writeFile(String pathFile, boolean isContinuing, String content) throws Exception
{
		if (!FileUltis.isFileExists(pathFile)) {
			throw new Exception(FILE_NOT_EXIST);
		}
		FileOutputStream out = new FileOutputStream(pathFile, isContinuing);
		out.write(content.getBytes());
		out.close();
		System.out.println(WRITE_FILE_SUCCESS);
	}
```

Question 3: Tạo method write Object
Hãy viết 1 method để có thể write object
Chú ý:
Nếu object = null thì sẽ throw ra Exception "Error! Object is Null."
Nếu folder không tồn tại thì sẽ tạo folder.
Nếu file đã tồn tại trong folder thì sẽ ghi đè lên file cũ
Gợi ý: Tạo method **void writeObject(Object object, String path, String fileName)**

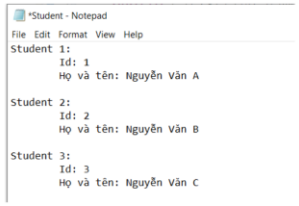
```
public static void writeObject(Object object, String path, String fileName) throws Exception {
		if (!FileUltis.isFileExists(path)) {
			throw new Exception(FILE_NOT_EXIST);
		}
		System.out.print("File name:");
		fileName = ScannerUltis.inputString();
		path = path + "\\" + fileName;
		FileOutputStream out = new FileOutputStream(path);
		ObjectOutputStream objectOut = new ObjectOutputStream(out);
		objectOut.writeObject(object);
		out.close();
		objectOut.close();
		System.out.println(WRITE_FILE_SUCCESS);
	}
```

Question 4: Tạo method read Object
Tạo 1 method có đầu vào là String filePath và return ra Object
Chú ý:
Nếu file không tồn tại thì sẽ throw ra Exception "Error! File Not Exist." (sử dụng FileManager ở trên để check)
Gợi ý: Tạo method **Object readObject(String filePath)**

```
public static void writeObject(Object object, String path) throws Exception {
		if (!FileUltis.isFileExists(path)) {
			throw new Exception(FILE_NOT_EXIST);
		}
		FileOutputStream out = new FileOutputStream(path);
		ObjectOutputStream objectOut = new ObjectOutputStream(out);
		objectOut.writeObject(object);
		out.close();
		objectOut.close();
		System.out.println(WRITE_FILE_SUCCESS);
	}
```

Exercise 5: Demo File & IO Stream

Question 1: Tạo class Student có property id, name.
a) Sau đó khởi 3 instance từ console (sử dụng ScannerUtils)
b) Write 3 student này ra file tên là StudentInformation.txt
c) Sau đó đọc thông tin file StudentInformation.txt và in ra màn hình



Question 2: Tạo LogUtils (sử dụng write Object Exception)
Tạo 1 custom Exception Object, sau đó tại hàm main sẽ bắt exception ko mong muốn và lưu thông tin Exception.ser file
Các bước làm sẽ như sau:
B1: Tạo 1 object MyException sẽ extends Exception, trong MyException sẽ chứa các thông tin như: message, reason, StackTrace, time (thời gian bị Exception), …
B2: Sau đó tạo class LogUtils có 1 vài method như sau:
1. void writeLog (String message, String reason, StackTrace, Datetime time) (gợi ý: sử dụng IOManager ở trên để write file)
2. void writeLog (MyException exception)
B3: đặt try catch ở method main() hoặc ở bất kỳ chỗ nào muốn lưu lại thông tin exception
B4: Tạo method để đọc thông tin trong file Exception và in ra màn hình
B5: Demo chương trình

Question 3 (Optional):
Thiết kế bài lô đề của exercise trước với những cải tiến về handling exception, đăt các property static, final cho hợp lý và thêm các chức năng sau đây:
a) Đăng ký tài khoản (thông tin các tài khoản sẽ lưu vào file)
Có 2 loại tài khoản admin và user.
b) Đăng nhập vào hệ thống
c) Đăng xuất tài khoản
d) Cho phép nạp tiền vào tài khoản (lưu thông tin vào file)
e) Lưu thông tin kết quả lô đề từng ngày vào file
f) Lưu thông tin người dùng chơi lô đề vào file
g) Tạo thêm chức năng sau:
1) Chỉ có admin mới có thể xem thông tin tất cả các tài khoản (giống table ở SQL khi SELECT * Account, ẩn mật khẩu đi)
2) Người dùng có thể xem lại thông tin lô đề từng ngày (hệ thống có hỗ trợ chức năng tìm kiếm theo ngày, và lọc chỉ xem từ ngày nào tới ngày nào)