12주차 과제 및 실습

과제 및 실습

- **과제는 문제 및 프로그램 실습으로 나뉘어 있습니다.**
  - 반드시 실습 예제를 작성해보고 실행해봅니다.

- **제출 파일 : 보고서+소스코드**
  - 보고서 : 한글 또는 word 파일
  - 소스코드 : java 파일만 제출
  - 위 두 파일을 학번_이름_실습주차.zip으로 압축하여 제출

  (3개의 실습문제가 있다면 exec_2019111998_1.java, … exec_2019111998_3.java, 2019111998_이선호_2주차.hwp를 zip파일로 압축해서 제출)

- **보고서 내용**
  - 과제 문제 : 해답, 해답의 이유
  - 실습 문제 : 해답소스코드, 프로그램 설명, 결과 화면(결과 캡처)
    - 프로그램 설명 : 작성한 소스코드의 내용, 소스코드 내 주석으로 대체 가능
  - 보고서 파일명 : 학번_이름_실습주차.hwp
    - ex) 2023111010_홍길동_2주차.hwp

- **소스코드**
  - 파일명 : exec_학번_문제번호.java(2번 문제의 경우 exec_2019111998_2.java)
  - 클래스명이 문제에 지정된 경우 파일명은 문제에 지정된 클래스명으로 설정

- **제출 : eclass**
  - 과제 연장 제출을 희망할 경우 직접 또는 이메일을 통해 조교에게 요청

프로그램 예제1. 제네릭 예제(1)

- 예제 7-9 제네릭 스택 만들기
- 맴버 ArrayList를 이용하기

```
Korea
Busan
Seoul
```

```java
import java.util.ArrayList;

class GStack<T>{
    public int tos;
    ArrayList<T> stack;

    public GStack() {
        tos = 0;
        stack = new ArrayList<T>();
    }
    public void push(T item) {
        stack.add(item);
        tos++;
    }
    public T pop() {
        if(tos > 0) {
            tos--;
            return (T)stack.remove(tos);
        }else
            return null;
    }
}

public class MyStack {
    public static void main(String[] args) {
        GStack<String> myStack = new GStack<String>();
        myStack.push("Seoul");
        myStack.push("Busan");
        myStack.push("Korea");

        for(int i = 0; i < 3; i++)
            System.out.println(myStack.pop());
    }
}
```

프로그램 예제2. 제네릭 예제(2)

- 제네릭 예제(1)에서 제네릭 메소드 추가

```
import java.util.ArrayList;

class GStack<T>{
    public int tos;
    ArrayList<T> stack;

    public GStack() {
        tos = 0;
        stack = new ArrayList<T>();
    }
    public void push(T item) {
        stack.add(item);
        tos++;
    }
    public T pop() {
        if(tos > 0) {
            tos--;
            return (T)stack.remove(tos);
        }else
            return null;
    }
}

class ReverseStack{
    public static <T> GStack<T> reverse(GStack<T> s){
        GStack<T> result = new GStack<T>();
        while(true) {
            T tmp;
            tmp = s.pop();
            if(tmp != null)
                result.push(tmp);
            else
                break;
        }
        return result;
    }
}
```

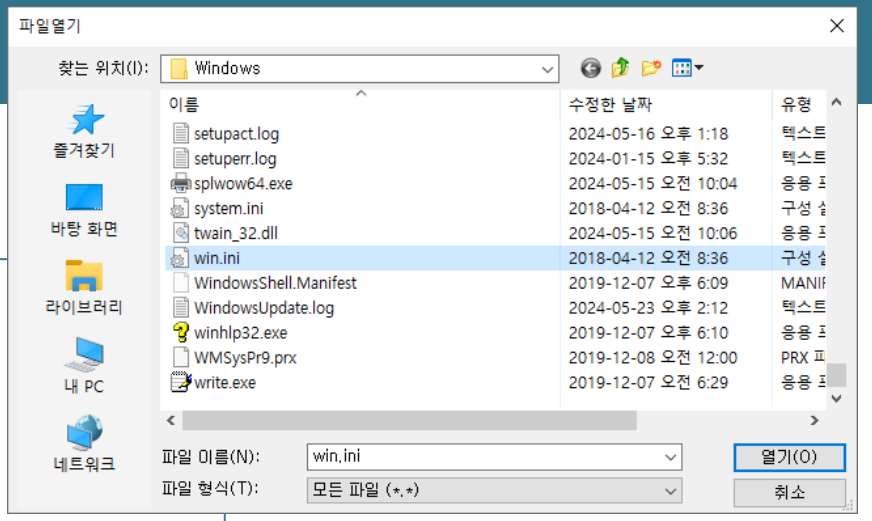
```
public class MyStack {
    public static void main(String[] args) {
        GStack<String> myStack = new GStack<String>();
        myStack.push("Seoul");
        myStack.push("Busan");
        myStack.push("Korea");

        GStack<String> newStack = ReverseStack.reverse(myStack);

        for(int i = 0; i < 3; i++)
            System.out.println(newStack.pop());
    }
}
```

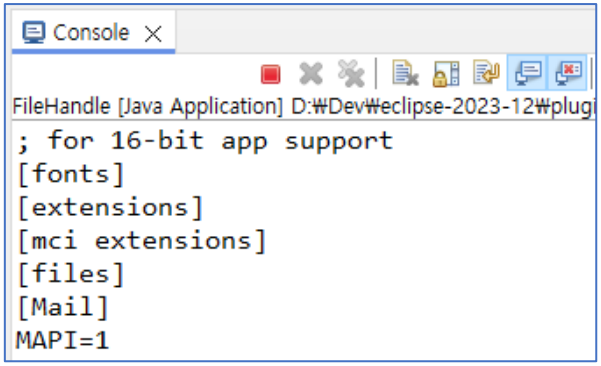
```
Seoul
Busan
Korea
```

프로그램 예제3. File 입출력

- File 을 읽고 다른 File에 쓰기

```
import java.awt.*;
import java.io.*;

public class FileHandle {
    public static void main(String[] args) {
        FileReader fr = null;
        FileWriter fw = null;

        Frame frame = new Frame("Parent");
        FileDialog fd = new FileDialog(frame, "파일열기", FileDialog.LOAD);
        fd.setVisible(true);     //대화상자로 읽을 파일 선택

        String path = fd.getDirectory();
        String name = fd.getFile();

        try {
            fr = new FileReader(path+name);
            fw = new FileWriter("d:\\out.txt");

            int c;

            while( ( c = fr.read() ) != -1){
                System.out.print((char) c);
                fw.write((char)c);
            }
            fr.close();
            fw.close();      //닫지 않으면 파일의 쓰기가 완료되지 않는다.
        }catch(IOException e) {
            System.out.println("입출력 오류");
        }
    }
}
```





과제문제 : 다음의 문제를 풀고 정답의 이유도 함께 작성하세요.

1. 제네릭에 대한 설명으로 옳지 않은 것은?

① 제네릭은 사용자의 클래스에 타입 매개 변수를 사용하여 일반화한 클래스이다.

② 제네릭 클래스의 내부에 여러 개의 값을 저장하기 위해 타입 매개변수를 이용하여 멤버 변수의 배열을 생성할 수 있다.

③ 제네릭 클래스의 내부에 타입 매개 변수를 이용한 매개변수(객체)를 생성할 수 있다.

④ 제네릭의 타입 매개 변수에 타입은 컴파일 시에 타입이 결정되므로 런타임 오류를 방지할 수 있다.

⑤ 제네릭 메소드를 호출할 때 타입 매개 변수의 타입을 지정하지 않아도 컴파일러가 자동으로 타입을 추정한다.

2. 스트림에 대한 설명으로 틀린 것은?

① 입출력이 동시에 되는 스트림은 없다.

② 스트림은 다른 스트림과 연결될 수 없다.

③ 스트림은 바이트를 다루는 스트림과 문자만 다루는 스트림으로 나누어진다.

④ 스트림은 먼저 들어온 데이터를 먼저 보내는 방식으로 동작한다.

과제문제 : 다음의 문제를 풀고 정답의 이유도 함께 작성하세요.

3. 다음 코드를 포함하는 자바 프로그램을 작성하고 결과를 캡처하라. 또한 각 메소드의 의미를 설명하라.

```
File file = new File("C:\\Windows\\system.ini");
```

① file.isFile()의 리턴값 출력

② file.getParent()의 리턴값 출력

③ file.getPath()의 리턴값 출력

④ file.getName()의 리턴값 출력

⑤ file.exist()의 리턴값 출력

실습 문제1

Exec1) 제네릭을 이용하여 Queue를 만들어라.

- MyQueue라는 이름으로 제네릭 클래스를 만든다.
- MyQueue의 내부에는 LinkedList 컬렉션을 이용하여 데이터(객체)를 저장한다.
- enQueue()메서드를 이용하여 큐의 뒤에 데이터를 넣는다.
- deQueue()메서드를 이용하여 큐의 앞에 데이터를 삭제하고 가져온다.

- 실행 결과

```
public class MyQueueEx {
    public static void main(String[] args) {
        MyQueue<String> queue = new MyQueue<String>();
        queue.enQueue("Seoul");
        queue.enQueue("Busan");
        queue.enQueue("Korea");

        for(int i = 0; i < 3; i++)
            System.out.println(queue.deQueue());
    }
}
```

```
Seoul
Busan
Korea
```

실습 문제2

Exec2) 앞 문제의 Queue를 이용하여 수정된 프로그램을 작성하라.

- 위 문제의 MyQueue를 수정하기 위하여 프로젝트를 복사하고 이름을 변경하라.
- 위 문제의 MyQueue를 양방향 Queue로 변경한다.
- EnQueueFirst() 또는 EnQueueLast() 메서드를 만들어 큐의 앞 또는 뒤의 데이터를 넣는다.
- DeQueueFirst() 또는 DeQueueLast() 메서드를 만들어 큐의 앞 또는 뒤의 데이터를 삭제하고 가져온다.
- 큐 안의 데이터를 DeQueueXXX() 메서드로 삭제하지 않고 출력하는 printQueue() 메서드를 추가하라.
- Main 함수는 위 수정된 Queue를 테스트 및 출력 할 수 있는 적절한 코드를 작성한다.

실습 문제3

Exec3) 2개의 파일을 입력받고 비교하는 프로그램을 작성하라.

- Dialog를 순차적으로 열어 두 개의 파일을 선택한다.
- 두 파일을 내용을 비교하여 같으면 “두 파일의 내용은 동일합니다”를 출력한 후, 파일의 내용을 출력한다.
- 두 파일의 내용이 다르면 “두 파일의 내용은 다릅니다.”를 출력하고 마친다.

- 실행 결과

```
첫번째 파일 명 : C:\Windows\system.ini
두번째 파일 명 : D:\out.txt
-> 두 파일의 내용이 같습니다.

<< 파일 내용 >>

; for 16-bit app support
[386Enh]
woafont=dosapp.fon
EGA80WOA.FON=EGA80WOA.FON
EGA40WOA.FON=EGA40WOA.FON
CGA80WOA.FON=CGA80WOA.FON
CGA40WOA.FON=CGA40WOA.FON

[drivers]
wave=mmdrv.dll
timer=timer.drv

[mci]
```