스레드(thread) : 프로그램을 실행하는 하나의 실, 제어의 개념

⁄ 운영체제에서 태스크(작업)을 실행하고, 관리하는 단위

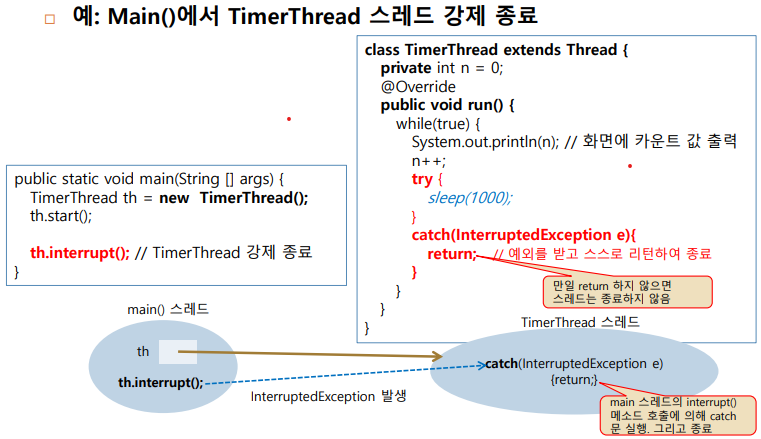
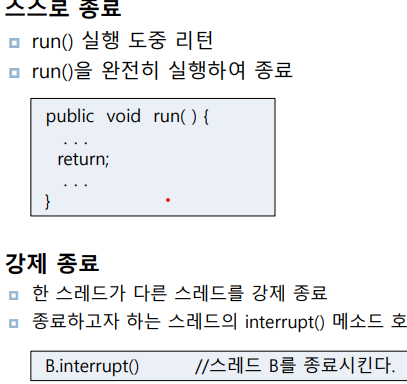
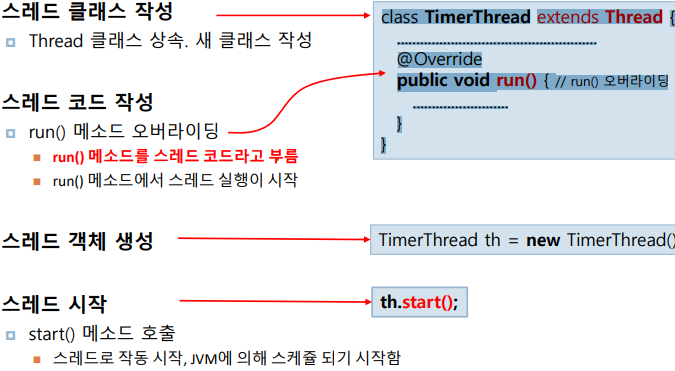
스레드를 생성, 중단, 종료시키고, 실행할 스레드를 스케쥴링 하는 권한은 운영체제에 있음

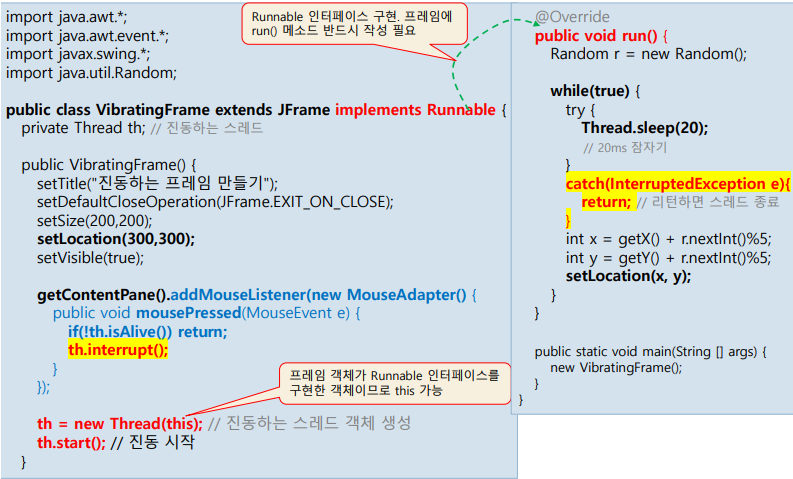
스레드의 생명 주기(실행, 멈춤, 종료 등)는 자바 가상 기계(JVM)에 의해 관리됨

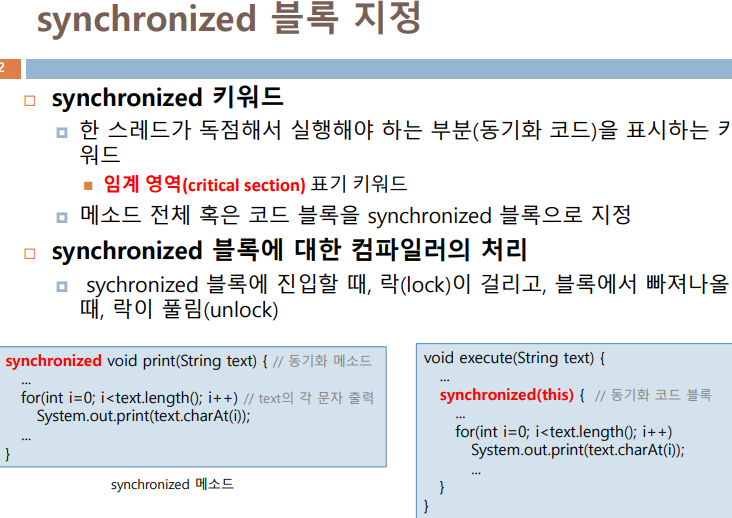
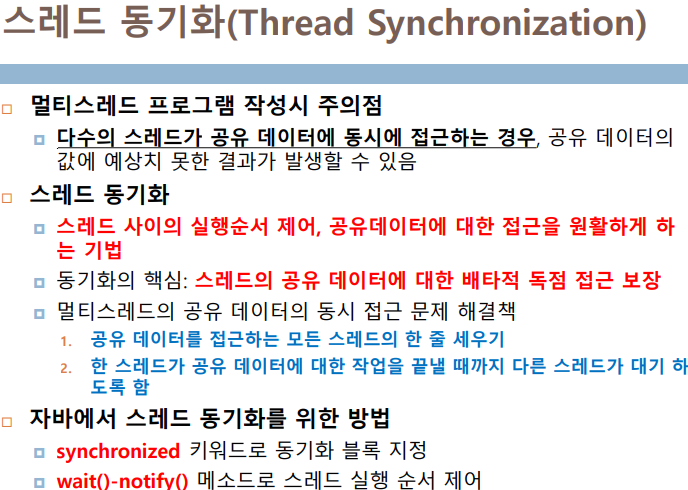
JVM과 멀티스레드의 관계 ⁄ 하나의 JVM은 하나의 자바 응용프로그램만 실행

ν 자바 응용프로그램이 시작될 때 JVM이 함께 실행됨

ν 자바 응용프로그램이 종료하면 JVM도 함께 종료함⁄ 하나의 응용프로그램은 하나 이상의 스레드로 구성 가능



12 -4



자바의 입출력 스트림

입출력 장치와 자바 응용 프로그램을 연결하는 소프트웨어 모듈

⁄ 입력 스트림 : 입력 장치로부터 **자바 프로그램으로 데이터를 전달**하는 객체

⁄ 출력 스트림 : 자바 프로그램에서 **출력 장치로 데이터를 보내는** 객체

♦ 특징 ⁄ 입출력 스트림 기본 단위 : 바이트나 문자 ⁄ 단방향 스트림, 선입선출 구조

자바 프로그램 개발자는 데이터를 직접 입력 장치에서 읽지 않고 입력 스트림을 통해 읽으며, 스크린 등 출력 장치에 직접출력하지 않고 출력 스트림에 출력한다

문자 스트림 ⁄ 문자만 입출력 하는 스트림 (자바는 1문자는 2bytes)

⁄ 문자가 아닌 **바이너리 데이터는 처리하지 못함**

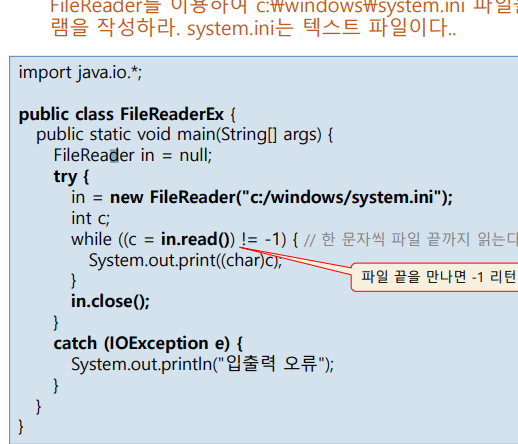
⁄ 문자가 아닌 데이터를 문자 스트림으로 출력하면 깨진 기호가 출력 ⁄ 바이너리 파일을 문자 스트림으로 읽으면 읽을 수 없는 바이트가 생겨서 오류 발생

⁄ 예) 텍스트 파일을 읽는 입력 스트림

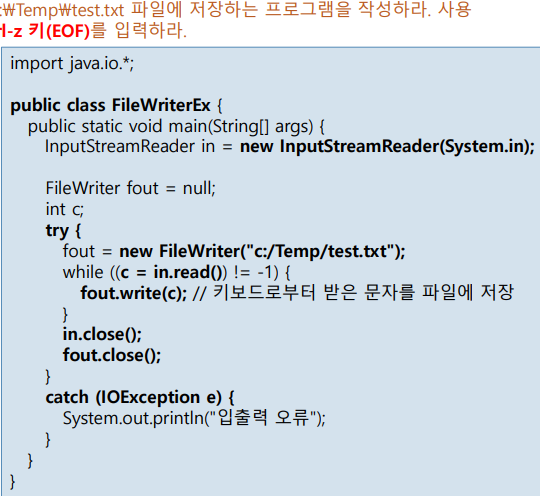
바이트 스트림 ⁄ 입출력 데이터를 단순 바이트의 흐름으로 처리

⁄ 문자 데이터 ∙ 바이너리 데이터 상관없이 처리 가능

⁄ 예) 바이너리 파일을 읽는 입력 스트림

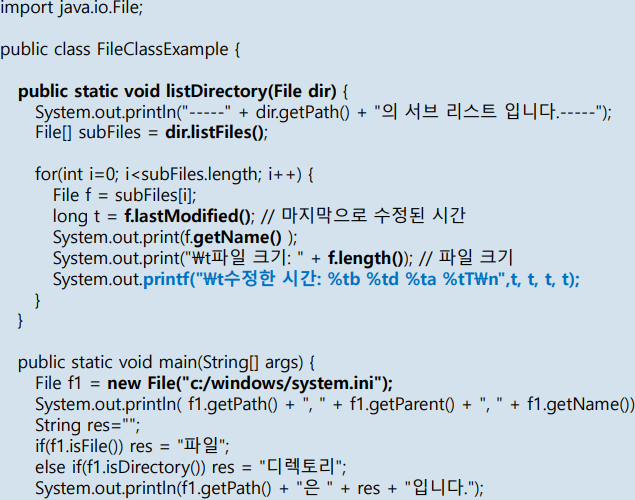
13-1

: FileReader로 텍스트 파일 읽기

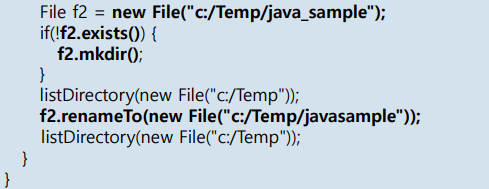


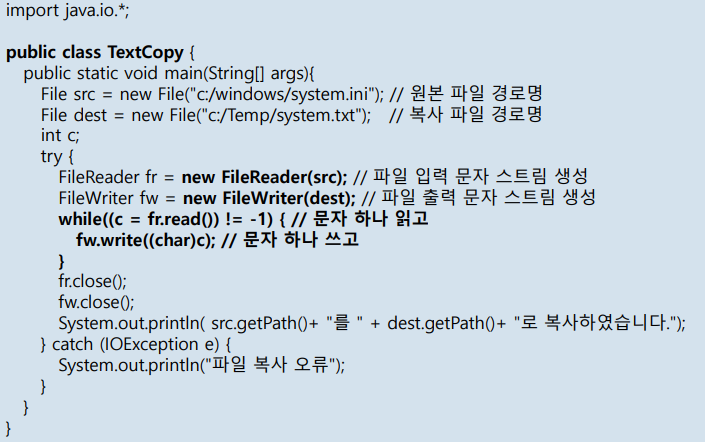
2

FileWriter를 이용하여 텍스트 파일 쓰기

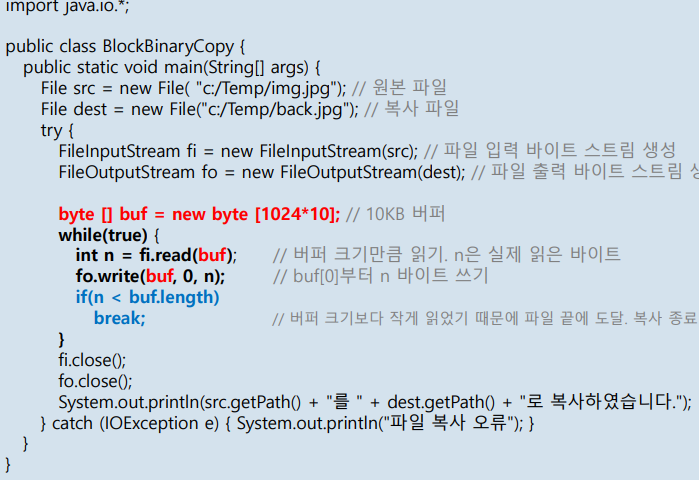


13-5 : File 클래스를 활용한 파일 관리



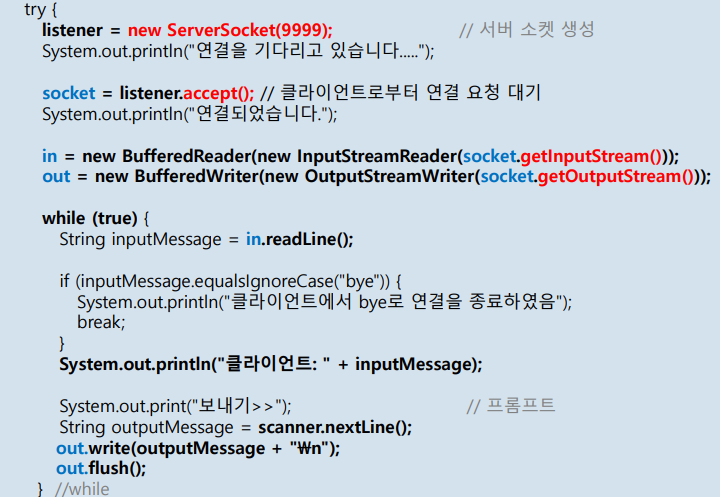
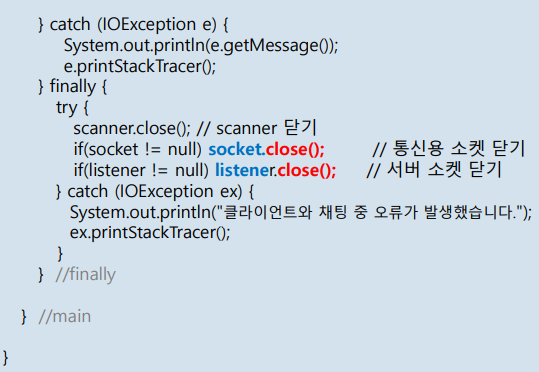


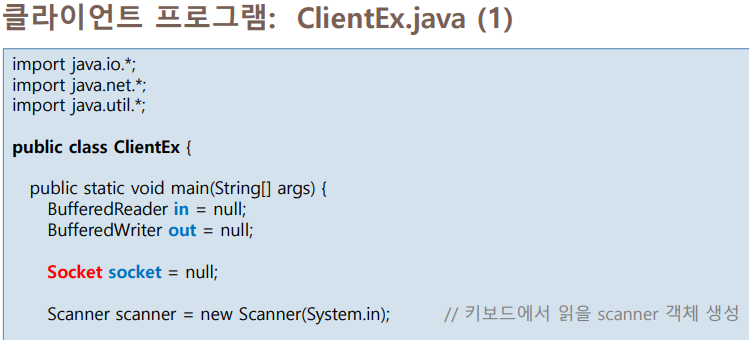
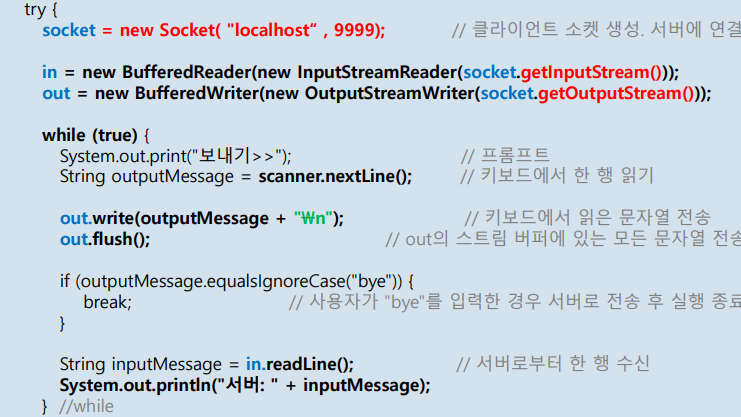
13-6 : 텍스트 파일 복사

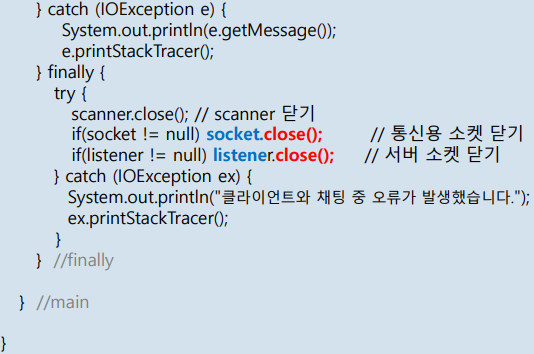


13-8 : 고속 복사를 위한 블록 단위 바이너리 파일 복사

바이너리버퍼카피





서블릿(Servlet) : 자바를 이용해 웹 애플리케이션을 만들기 위한 서버 측 프로그램.

init(ServletConfig config)

서블릿 초기화 메서드. 서블릿이 처음 생성될 때 한 번 호출됩니다.

주로 초기화 작업을 수행.

service(ServletRequest req, ServletResponse res)

클라이언트 요청을 처리하는 메서드.

HttpServlet 클래스에서는 doGet()이나 doPost() 같은 메서드로 분기.

doGet(HttpServletRequest req, HttpServletResponse res)

GET 요청을 처리.

URL 파라미터를 읽고, 서버 자원을 조회하거나 HTML 폼 데이터를 처리.

doPost(HttpServletRequest req, HttpServletResponse res)

POST 요청을 처리.

주로 폼 제출 데이터를 처리하거나 데이터베이스에 정보를 저장.

destroy()

서블릿 종료 메서드. 서블릿이 제거되기 전에 호출됩니다.

리소스 해제 작업을 수행.

getServletConfig()

서블릿 초기화 시 전달된 ServletConfig 객체를 반환.

초기화 파라미터에 접근할 수 있습니다.

getServletContext()

서블릿이 속한 웹 애플리케이션의 ServletContext 객체를 반환.

애플리케이션 범위에서 공유되는 리소스나 데이터를 관리.

자바빈즈(JavaBeans) : 재사용 가능한 소프트웨어 컴포넌트.

getPropertyName()

프로퍼티 값을 반환하는 getter 메서드.

예: public String getName() { return name; }.

setPropertyName(Type value)

프로퍼티 값을 설정하는 setter 메서드.

예: public void setName(String name) { this.name = name; }.

public BeanClass()

기본 생성자. 인자가 없는 생성자입니다.

JDBC (Java Database Connectivity) : 자바에서 데이터베이스에 접근하기 위한 API.

DriverManager.getConnection(String url, String user, String password)

데이터베이스 연결을 획득.

Connection conn = DriverManager.getConnection(dbURL, user, password);.

Connection.createStatement()

SQL 문을 실행하기 위한 Statement 객체 생성.

Statement stmt = conn.createStatement();.

Statement.executeQuery(String sql)

SELECT SQL 문 실행, 결과 집합 반환.

ResultSet rs = stmt.executeQuery("SELECT \* FROM users");.

Statement.executeUpdate(String sql)

INSERT, UPDATE, DELETE SQL 문 실행, 변경된 행 수 반환.

int rows = stmt.executeUpdate("UPDATE users SET name='John' WHERE id=1");.

PreparedStatement

SQL 문을 미리 컴파일하여 실행. 성능 향상과 SQL 인젝션 방지.

String sql = "INSERT INTO users (name, email) VALUES (?, ?)"; PreparedStatement pstmt = conn.prepareStatement(sql); pstmt.setString(1, "John"); pstmt.setString(2, "john@example.com"); int rows = pstmt.executeUpdate();

ResultSet.next()

결과 집합에서 다음 행으로 이동.

while (rs.next()) { String name = rs.getString("name"); }

Connection.close()

데이터베이스 연결 종료. 예: conn.close();.

세션(Session) : 클라이언트와 서버 간의 상태 정보를 유지하기 위한 방법.

HttpSession getSession()

현재 요청과 연관된 세션을 반환. 세션이 없으면 새로 생성.

예: HttpSession session = request.getSession();.

HttpSession getSession(boolean create)

현재 요청과 연관된 세션을 반환. create가 true이면 세션이 없을 경우 새로 생성.

예: HttpSession session = request.getSession(false);.

void setAttribute(String name, Object value)

세션에 속성 설정.

예: session.setAttribute("username", "John");.

Object getAttribute(String name)

세션에서 속성 값 가져오기.

예: String username = (String) session.getAttribute("username");.

void removeAttribute(String name)  
세션에서 속성 제거.

예: session.removeAttribute("username");.

void invalidate()

세션 무효화.

예: session.invalidate();.