BLUETOOTH SERVICE

2020년도 1학기 앱 프로그래밍 11주차 수업

목 차

1~3교시

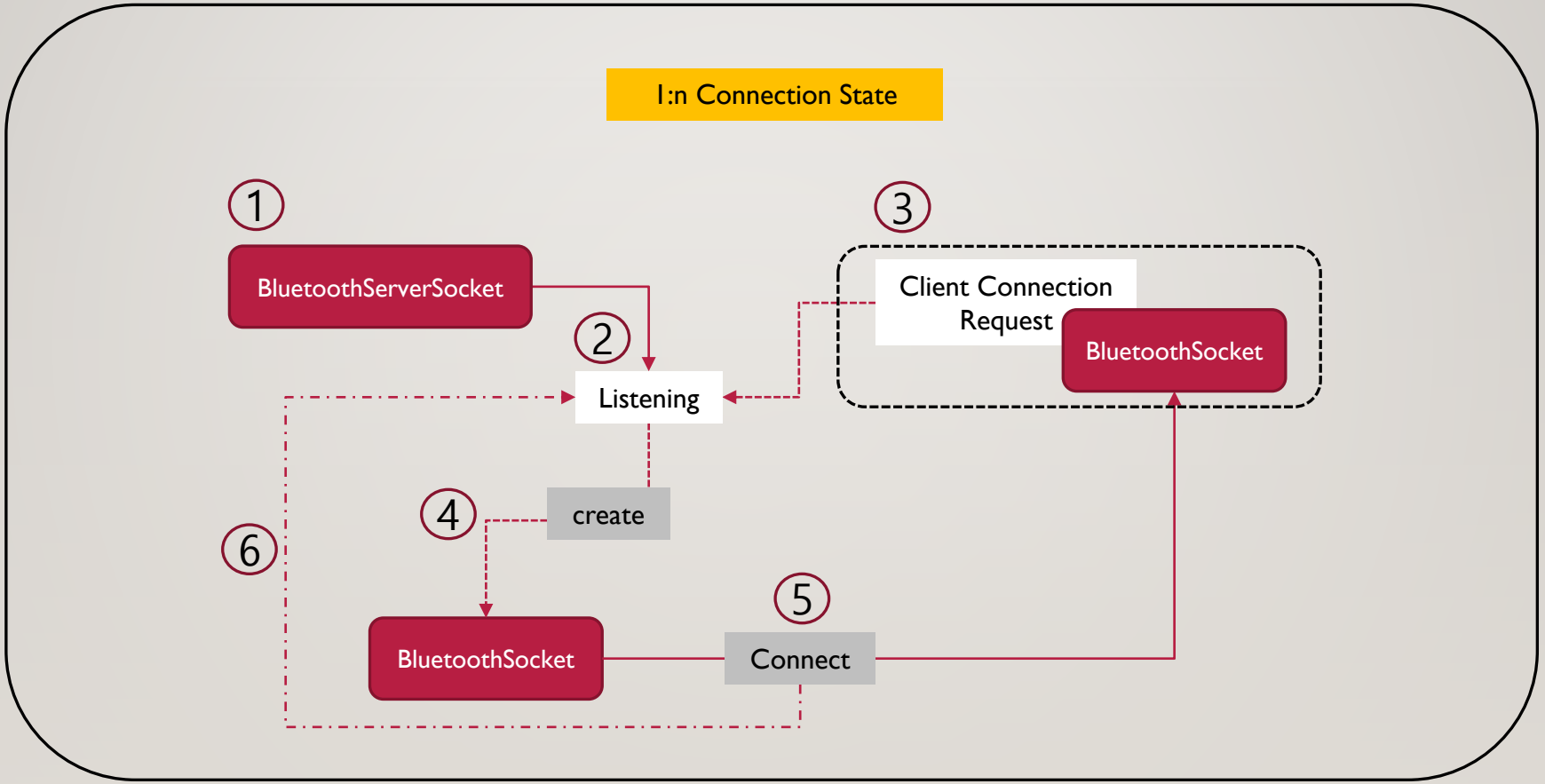
2020년도 1학기 앱 프로그래밍

– BLUETOOTH CONNECTION & SEND / RECEIVE DATA

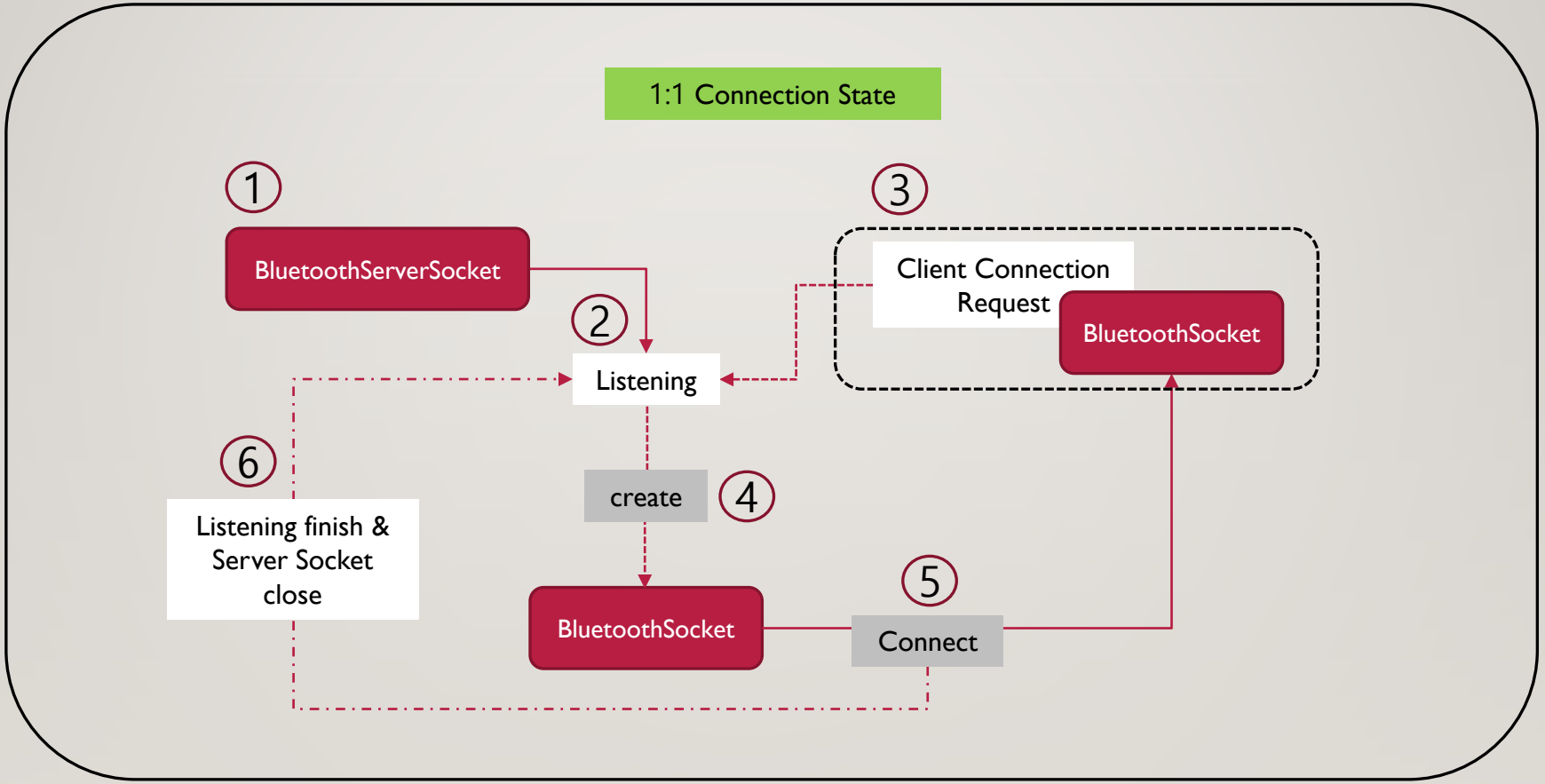
I. 블루투스 서버 소켓

- 블루투스 장치 연결
  - 클래스 : BluetoothServerSocket 제공
  - 서버 소켓 목적 : 들어오는 연결 요청 수신 대기 / 연결 수락(BluetoothSocket) 제공
  - 1:1 연결
    - BluetoothServerSocket 리스닝 → 요청 수신 → BluetoothSocket 생성(연결) → 리스닝 취소
  - 1:n 연결
    - BluetoothServerSocket 리스닝 → 요청 수신 → BluetoothSocket 생성(연결) → 리스닝 반복

1. 블루투스 서버 소켓 (계속)



I. 블루투스 서버 소켓 (계속)



1. 블루투스 서버 소켓 (계속)

- 서버 소켓 설정 과정
  1. listenUsingRfcommWithServiceRecord() 호출 → BluetoothServerSocket 반환
     - SDP(Service Discovery Protocol) DB에 해당 항목 기록
     - UUID → SDP 항목에 포함, 클라이언트 기기와 연결 동의 판단 기준
       - 클라이언트가 연결하고자 하는 서비스의 고유 식별 ID
       - UUID가 일치해야 연결이 수락됨
       - UUID 생성기 사용 → fromString(String)으로 초기화
  2. accept()호출 → 연결 요청에 대한 수신 대기 시작
     - 차단 호출이며 연결 수락 또는 예외 발생시 값 반환
     - UUID 일치 연결 요청만 수락
     - 연결 성공 시 accept() 메서드가 연결된 BluetoothSocket 인스턴스 반환
  3. close()호출 → 추가 연결을 수락하지 않고 1:1 통신인 경우
     - BluetoothServerSocket이 닫히고 accept()로 연결된 BluetoothSocket 은 유지

I. 블루투스 서버 소켓 (계속)

- 서버 소켓 설정 추가
  - 블루투스는 TCP/IP와 달리 RFCOMM이며 한번에 채널당 단일 연결만 허용
  - 대부분의 경우 연결된 소켓 수락 직후 BluetoothServerSocket에 close() 호출
  - 차단 호출 → 연결 요청 수신 전까지 대기 상태 유지
    - UI 스레드에서 실행하면 안됨 → 앱의 UI 등 상호 작용에 문제 발생 가능 때문
    - Thread로 보호된 별도의 작업으로 실행

2. 블루투스 클라이언트 소켓

- 리스닝 상태 서버 소켓으로의 연결
  - 원격 서버 소켓으로의 접속 요청을 뜻함
  - 서버 소켓에서 연결 수락 상태의 연결 과정에 BluetoothDevice 객체 사용
  - BluetoothDevice로 BluetoothSocket 획득 후 연결 시작

- 블루투스 소켓 생성 기본 과정
  1. BluetoothDevice 의 멤버 createRfcommSocketToServiceRecord(UUID) 호출
     - 초기화 된 BluetoothSocket 객체 반환
     - UUID 매개변수는 서버 소켓에서 사용하는 UUID이며 일치하는 UUID로 연결 시도
  2. connect() 메서드 호출로 연결 시작 → 차단 호출!
     - 차단 호출이므로 반드시 Activity UI 스레드와 분리된 스레드로 수행
       - 기기 검색 중이라면 cancelDiscovery()호출하여 검색 중지 필요

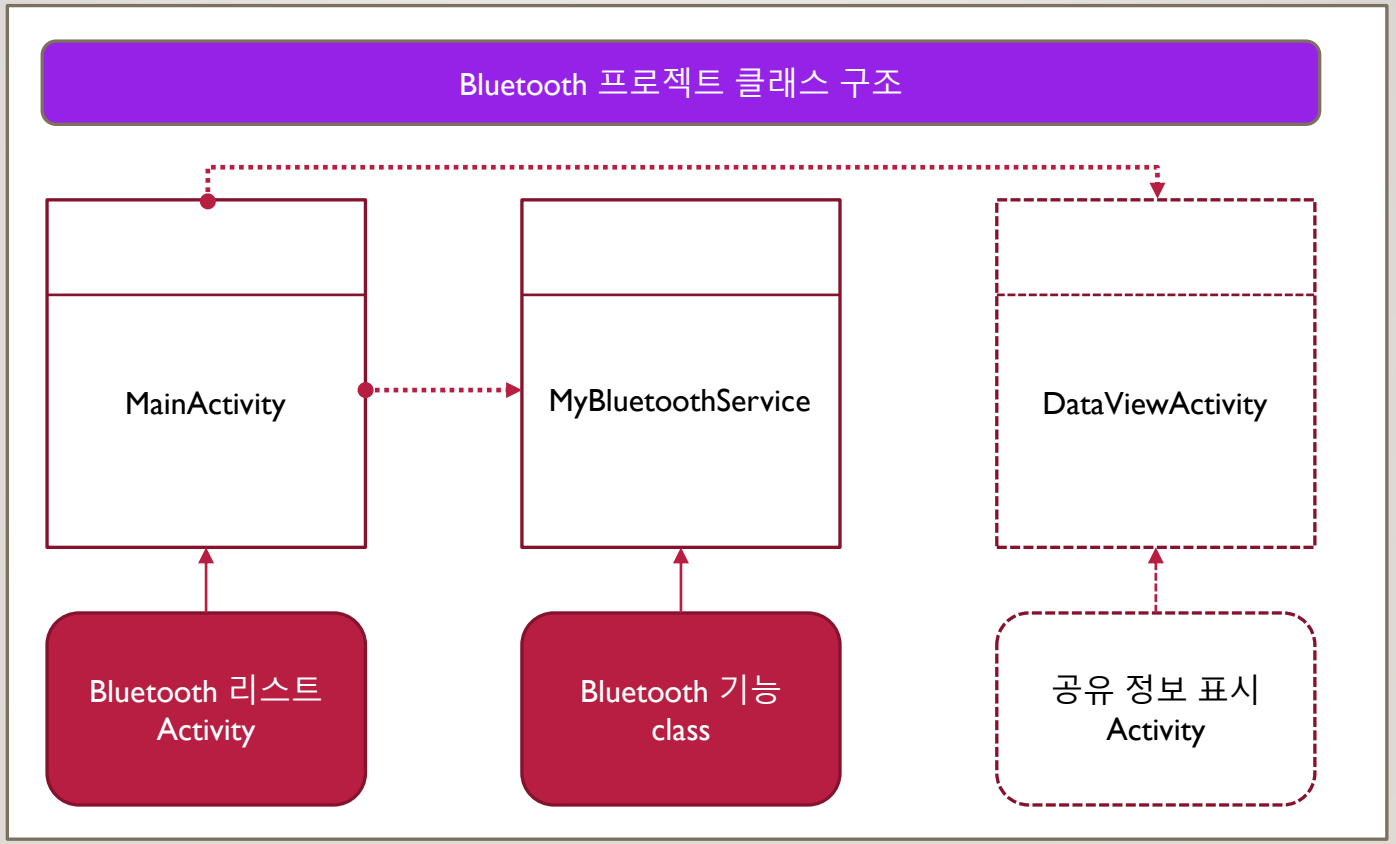
3. 블루투스 연결 관리

- 연결 성공 시 연결된 BluetoothSocket 관리
  - 기기간 정보 공유(송수신) 가능 상태
  - 데이터 수신 및 전송을 위한 스레드로 구성

- 데이터 전송 절차
  - 데이터 수신을 위한 getInputStream() 메서드 → InputStream 객체 얻기
  - 데이터 송신을 위한 getOutputStream() 메서드 → OutputStream 객체 얻기
  - 차단 호출 메서드 사용 → 별도의 스레드로 구성 요구
    - read(byte[]) 메서드로 수신 데이터 읽기 수행
      - 스레드 run() 의 기본 루프로 읽기 동작 처리 →자동 수신처리 미 사용시 중간 버퍼 오버플로우 발생
    - write(byte[]) 메서드로 송신 데이터 쓰기 수행
      - 스레드 개별 메서드 사용 OutputStream 에 쓰기 동작 처리

4. 프로젝트 구조

- 프로젝트의 핵심 클래스 구조



4. 프로젝트 구조 (계속)

- MyBluetoothService 클래스 속성

```
// 서버 소켓 생서시 SDP 기록을 위한 이름
private static final String NAME_SECURE = "BluetoothChatSecure";

// 연결을 위한 고유한 UUID 값들
private static final UUID MY_UUID_SECURE = UUID.fromString("fa87c0d0-afac-11de-8a39-0800200c9a66");

// 멤버 필드
private final BluetoothAdapter mAdapter;
private int mState;
private  final Handler mHandler;
private AcceptThread mAcceptThread;
private ConnectedThread mConnectedThread;
private ConnectThread mConnectThread;

// 현재 연결 상태를 나타내는 상수 값들
public static final int STATE_NONE = 0;       // we're doing nothing
public static final int STATE_LISTEN = 1;     // 연결 요청을 기다리며 리스닝하는 상태
public static final int STATE_CONNECTING = 2; // 외부 연결을 진행하는 상태
public static final int STATE_CONNECTED = 3;  // 원결 장치와 연결된 상태
```

4. 프로젝트 구조 (계속)

- MyBluetoothService 클래스 메서드

```
// 블루투스 서비스 기능 클래스 생성자
public MyBluetoothService(Context context, Handler handler){...}

// 서비스 상태 반환 메서드
public synchronized int getState() { return mState; }

// 리스트 activity 에서 호출 : 원격장치 연결시 또는 탐색 가능 모드 진입시 호출
public synchronized void start(){...}

// 모든 스레드 종료 처리
public synchronized void stop() {...}

public synchronized void manageMyConnectedSocket(BluetoothSocket socket, BluetoothDevice device) {...}

// 서버 소켓으로 연결을 시도하는 작업 시작 메서드
public synchronized void connect(BluetoothDevice device){...}

// 송신 메서드 : UI activity에서 메시지 송신시 호출
public void write(byte[] out) {...}
```

4. 프로젝트 구조 (계속)

- MyBluetoothService 클래스 내장 스레드 클래스

```
// 서버 소켓을 통한 연결 요청 대기와 연결 수신시 처리 스레드
private class AcceptThread extends Thread {...}

// 연결된 스레드로 데이터 송 수신 처리 스레드
private class ConnectedThread extends Thread{...}

// 원격 장치로 연결을 수행하고자 할때 사용하는 스레드
private class ConnectThread extends Thread {...}
```

4. 프로젝트 구조 (계속)

- AcceptThread 스레드 클래스 → 연결 요청 수신 및 수락

```
// 서버 소켓을 통한 연결 요청 대기와 연결 수신시 처리 스레드
private class AcceptThread extends Thread {
    // 서버 소켓
    private final BluetoothServerSocket mmServerSocket;

    // 리스닝 소켓 생성 작업 처리
    public AcceptThread() {...}

    // 연결 요청 대기 및 연결 소켓 작업 처리
    @Override
    public void run() {...}

    // 리스닝 중지 및 서버 소켓 종료 처리
    public void cancel(){...}
}
```

4. 프로젝트 구조 (계속)

- ConnectThread 스레드 클래스 → 원격 기기 연결 요청

```
// 원격 장치로 연결을 수행하고자 할때 사용하는 스레드
private class ConnectThread extends Thread {
    private final BluetoothSocket mmSocket;
    private final BluetoothDevice mmDevice;

    // UUID로 원격 장치와 연결 소켓 생성 작업 처리
    public ConnectThread(BluetoothDevice device) {...}

    // 주변 장치 검색 중지, 원격 기기와 소켓을 통한 연결
    @Override
    public void run() {...}

    // 연결 종료 작업 처리
    public void cancel() {...}
}
```

4. 프로젝트 구조 (계속)

- ConnectedThread 스레드 클래스 → 연결된 기기 관리 (정보 공유)

```
// 연결된 스레드로 데이터 송 수신 처리 스레드
private class ConnectedThread extends Thread{
    private final BluetoothSocket mmSocket;
    private final InputStream mmInStream;
    private final OutputStream mmOutStream;

    // 연결된 기기의 입출력 스트림 생성 작업 처리
    public  ConnectedThread(BluetoothSocket socket){...}

    // 수신 데이터 읽기 작업 처리
    @Override
    public void run() {...}

    // 송신 작업을 위한 별도의 메서드
    public void write(byte[] buffer) {...}

    // 연결된 소켓 종료 작업 처리
    public void cancel() {...}
}
```

4. 앱 프로그래밍 학습을 위한 실습 (과제)

- 실습 및 예습 문제
  - 수업 시간에 설명한 블루투스 기능 클래스 구현
    - 블루투스 장치 연결을 위한 기능 클래스 구현
  - 질문 사항 있을 경우 단톡방을 통해 문의하세요!

- 제출 방법
  - http://ctl.gtec.ac.kr 의 과제 제출란에 제출
  - Java 파일과 xml 파일을 제출한다.
  - 압축파일은 자신의 "반_학번_이름.zip" 로 한다.
  - 제출 일자와 시간을 엄수하여 제출하세요!