**SPGP 중간고사 정리**

**- OOP**

1. Attribute = data = member variables = state = field
2. Behavior = operation = member function = method
3. 클래스 = 개념 = 타입
4. 객체 = 실체 = 변수
5. 캡슐화 : 데이터와 데이터를 다루는 방법을 묶는 것 (추상화)
6. 상속 : 하나의 클래스가지고 있는 특성들을 그대로 다른 클래스가 물려 받는 것
7. Is a 상속관계 : Poodle is A Dog
8. Has a 포함관계는 상속이 아니라 멤버로(구성으로) 갖는다 (사람은 심장을 갖고 있다)
9. 다형성 : 같은 명령을 다르게 구현이 가능 (포유류에게 번식해라 오리 너구리 – 알을 낳음 고양이, 개 – 새끼를 낳음 ) 상위 계층은 어떻게 돌아가는지 모르고 명령만 한다

**- Kotlin vs Java : 두개를 섞어서 쓰는 것도 가능**

코틀린 혼용 시 apply plugin : kotlin-android 추가해주기

자바 - 메모리 : 포인터 개념이 없고 자동으로 메모리를 관리함, Garbage Collector가 사용하지않는 자원을 자동으로 반환

* OOP : 객체 지향 프로그래밍
* 멀티 스레드: thread 클래스와 Runnable 인터페이스를 이용해 멀티 스레드 구현
* 동적 로딩 : 필요한 시점에 구현한 클래스를 로딩 가능

코틀린 - Functional Programing : 객체지향 프로그래밍, 함수형 프로그래밍 모두 지원, 함수는 일급 객체임, 고차함수로 사용가능

- 모든 타입이 클래스 타입 : 기본형 타입마저 클래스로 존재

- Nullable : Null 할당이 가능한 타입을 선언할 수 있다, NPE예외를 컴파일 시점에서 미리 방지 가능

**- Package name / Bundle identifier 리버스 도메인**

1. 실제 존재하지않는 도메인이 아니여도 상관없음 다른사람과 구별을 위한 것

Ex) (kr.ac.kpu.game.s2017182016.gamename)

**- manifest파일 :**

1. <application으로 감싸짐
2. 권한 (사용자에게 민감한 정보에 엑세스)
3. configChanges : 가로세로 바뀔때, 다시실행시키지말것 (orientation|screenSize)
4. screenOrientation : landscape,portrait 가로 세로 고정

**- minimum sdk(API LEVEL):**

1. 몇 버전에서 실행할 수 있나 api 숫자가 높을수록 최신 기능을 사용할 수 있다
2. 반대로 숫자가 낮은 기기에서는 프로그램이 돌아가지 않는다

**- AVD(Android Virtual Device) 대신 실기기로 테스트시 개발용으로 만드는게 필요함 (환경설정 - 기기 정보 - 빌드 넘버 7번터치시 개발자용으로 전환됨)**

**- 프로젝트의 빌드를 위해 build.gradle파일을 이용함(빌드를 할때 필요한 속성을 정함**

1. 컴파일버전,
2. 버전네임 : 사용자에 노출되는 앱 버전 ( 첫번째 숫자는 중요한 기능이 바뀔때, 두번째 숫자는 마이너한 기능향상 3번째는 아주 사소한기능이거나 버그픽스, 4번째는 빌드넘버(커밋한 개수,날짜, 다 다름),
3. 버전코드 : 업데이트 비교 대상, 사용자에게 노출이안됨

**- 리소스 폴더**

1. res - mimap 의 dpi 는 configuration임,
2. 해상도별로 리소스를 준비할 수 있지만 원래 해상도대로 이미지를 로드함(bitmapfactory 옵션)

**- layout**

**-** activity\_main.xml configuration에 따라 준비할 수 있으나 관리는 매우 불편함, 가급적 최대한 하나만 가지고 할 수 있는게 좋음

**- string table**

- 안드로이드에서 문자열을 처리할 때 소스코드에서 처리하는 걸 권장하지 않음(다국어 처리위해) -> string.xml에서 string id별로 내용을 작성하고 소스코드에서 id를 사용 (언어별로 설정도 가능, values, values-ko)

- 실제코드는 R.string.~

**- 파일 이름**

윈도우 에서 파일이름내에 ‘-‘ 사용가능하나 자바 아이덴티파이어로는 "-,(", 자료형,첫번째글자 숫자 쓰지 못함 (스트링테이블도 동일)

**- layout 크기 :**

1. match\_parent: 부모의 크기에 맞춤,
2. wrap\_content:내용 크기에 맞춤,
3. @dimen/score\_text\_size : configuration에따라 바꿔주기 가능(숫자가들어가는 모든곳)

**- 단위 :**

1. sp(폰트크기비례)
2. px(실제 디바이스 점수 쓰지 마라),
3. dp(기기와 관계없이 비슷한 크기 보장) - 반드시 단위를 명시 해야함

**- findViewById :**

- 비용이 드므로 여러 번 하지 않고 생성자에서 한번만 한다

**- xml :**

- <>여는태그와 </>닫는 태그가 있다 태그 사이에 태그가 있을 수 있다. ,

- <>…</> == < …/>

**- hint**

- 사용자에게 안내할 때 사용

**- 패딩과 마진의 차이 :**

- **패딩**은 내 컨텐트에서 얼마나 제외할건지 / **마진**은 parent에서 배치 시 얼마나 제외할건지

**- garbage collection :**

1. 전부 검사를 하여 마크가 되있지 않은 객체를 날려버림
2. 언제 일어나는지 알 수 없음,
3. 이 시점에 시스템이 잠시 멈춤, 이것 때문에 게임루프에서 추가 삭제를 하는 건 좋지 않음

**- atan vs atan2 :** atan2 가 더정확함

**- canvas** 현재상태 저장 save() 다시 pop해서 되돌림 restore() (스택 구조)

**- 버튼을 눌럿을때 코드 가 실행되는 방법 4가지**

1. Button helloButton = findViewById(R.id.helloButton);

helloButton.setOnClickListener(helloButtonListener);

View.OnClickListener helloButtonListener = new View.OnClickListener() {

@Override

public void onClick(View view) {

버튼클릭시 실행할 내용(코드)

}

};

1. **(인터페이스 방법)**Main Activity 에서 implements View.OnClickListener 해준 뒤 setOnClickListner(this);

public void onClick(View view) {

TextView tv = findViewById(R.id.mainTextView);

tv.setText("Hello");

}

더 간단해짐

1. **(함수 사용방법)** xml에서 android:onClick="onBtnWorld" 이렇게 지정하고

MainActivity 에서

public void onBtnWorld(View view) {

mainTextView.setText("World");

}

만 해주면 됨 더욱 간단

1. (객체로 구현하는 방법)

Button.OnclickListener btn = new View.OnClickListener() {

Public void onClick(View v) {

if(v.getId() == R.id.btn\_1) 버튼 클릭시 구현할 코드

}

}

On Create 에서 findViewById(R.id.btn\_1).setOnClickListener(btn)

**- alt + enter :**

멤버 변수 , 로컬변수, 상수 만들기, 임포트

**- builder.setTitle**

(빌더 패턴이 가지는 장점: 생성자를많이 만들지 않아도됨, 내용물이 덜 채워진 상태로 생성되는걸 막음)

**- Choreographer**

Choreographer를 이용해 60초마다 불리도록(게임루프)V-Sync에 맞춰서 호출할 수 있음, 타이머로도 가능

**- Vector vs Array**

1. 벡터를 안 쓰고 어레이를 씀
2. 벡터가 더 안전하나 비효율적, 느림 - 계속 접근허락을 받아야하기 때문에