[시스템 분석 설계 포트폴리오]

20210996 조준영

Contents.

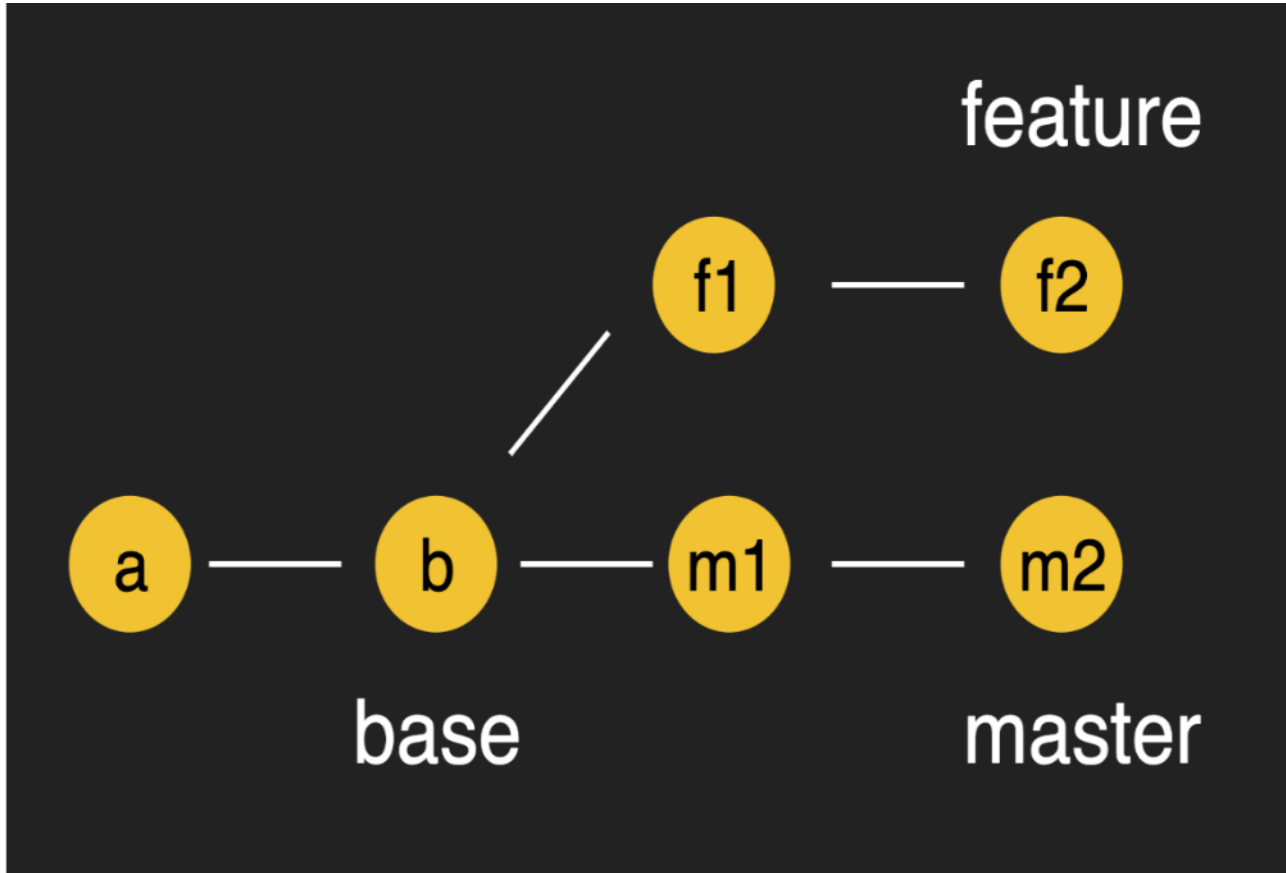
Chapter 1.

깃과 깃허브

깃의 기능 및 구조

1. 정의
   컴퓨터 파일의 변경을 추적하는 데 사용되는 버전 관리 시스템
2. 기능
   I. 소스 코드의 변경 사항을 추적하는 데 사용
   II. 소스 코드 관리에 분산 버전 제어 도구가 사용
   III. 여러 개발자가 함께 작업
   IV. 여러 개의 평행 분기를 통해 비선형 개발을 지원
3. 특징
   I. 기록 추적
   II. 백업 생성
   III. 협업 지원
   IV. 분산 개발
   V. 비선형 개발 지원

4. 브렌치 지원
   오픈 소스에 적합



깃의 기능 및 구조

1. Branch 개요

I. 개발과정에서 각각 서로 다른 버전의 코드가 만들어 질 수 있는 상황

II. 동일한 소스코드 위에서 어떤 개발자는 버그를 수정

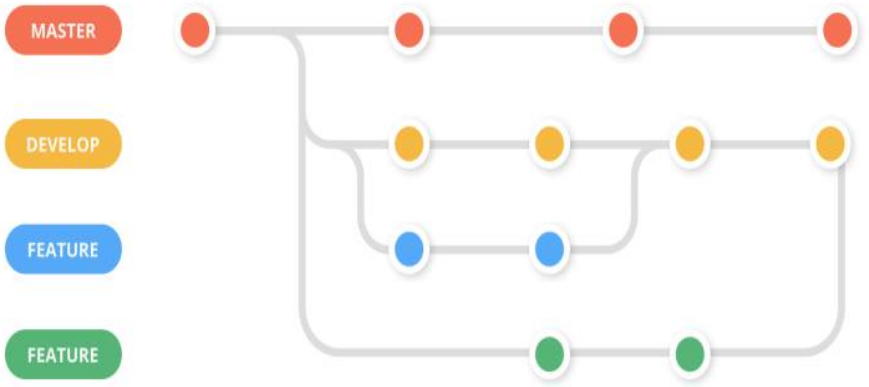
III. 또 다른 개발자는 새로운 기능을 만들어 냄

2. Branch

I. 여러 개발자들이 동시에 다양한 작업을 할 수 있게 만들어 주는 기능

II. 각자 독립적인 작업 영역 안에서 코드 변경

III. 분리된 작업 영역에서 변경된 내용은 나중에 원래의 버전과 비교하여 하나의 새로운 버전으로 생성 가능



깃의 기능 및 구조

Untracked

깃은 파일관리를 tracked와 untracked로 나누는데
untracked는 관리대상이 아닌 파일을 의미한다

tracked

깃은 파일관리를 tracked와 untracked로 나누는데
tracked는 관리대상인 파일을 의미한다
Working Directory와 Staging Area에 저장된다

Working Directory

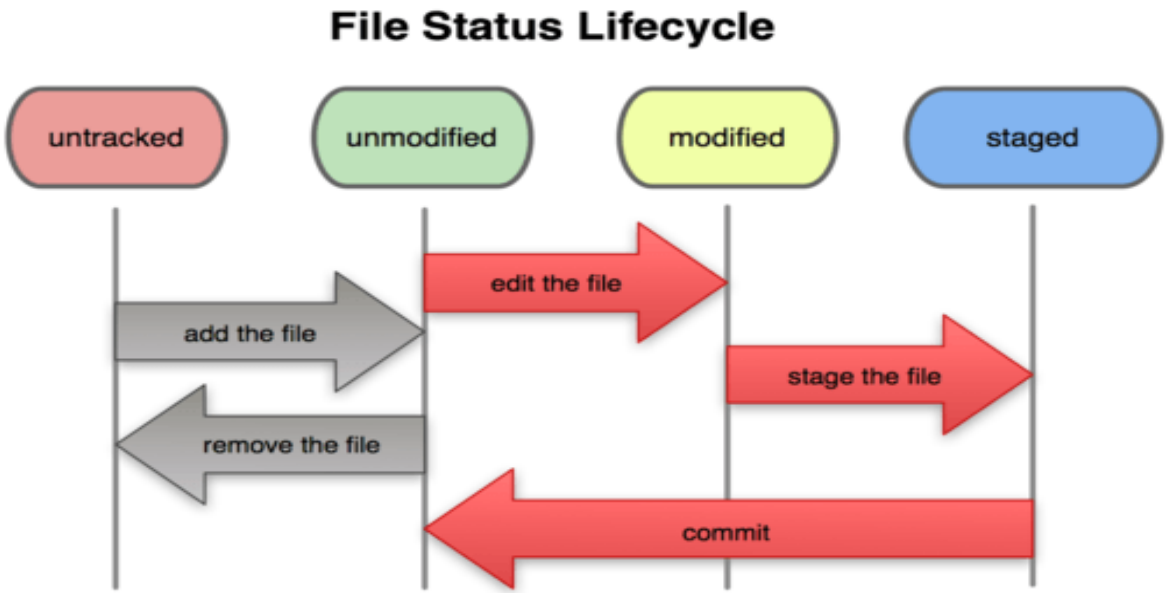
Working Directory는 현재 사용자가 사용하고 있는
컴퓨터에 작업 디렉토리를 말한다 각자의 컴퓨터에서
어떤 파일을 만들거나 파일의 내용을 수정하였거나
등등 각자 어떠한 작업을 한 디렉토리다

```
jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        github-basic/
        hello.py

nothing added to commit but untracked files present (use "git add" to track)
```

파일을 만든 후 깃의 상태

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   github-basic
        new file:   hello.py
```

Git add를 이용하여 파일을 추가한 상태



깃의 기능 및 구조

Staging Area

Staging Area 작업 진행 중에 commit 할 준비가 되어 있는 파일을 옮겨 두는 영역입니다 working directory에서 작업하고 commit할 대상만 선택적으로 옮겨 둘 수 있습니다

.git directory

git directory 는 전 단계인 staging area의 파일들을 최종적으로 우리 컴퓨터에 저장하는 저장소이다.

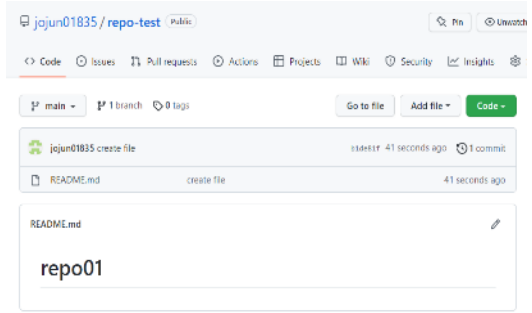
.git directory는 다음 단계인 remote repo에 저장하기 전 단계이다.remote repo는 원격 저장소라는 의미이고, github 저장소가 이에 해당한다.

```
jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial/git-repo (main)
$ git commit -m "first commit"
[main (root-commit) 099f5ff] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

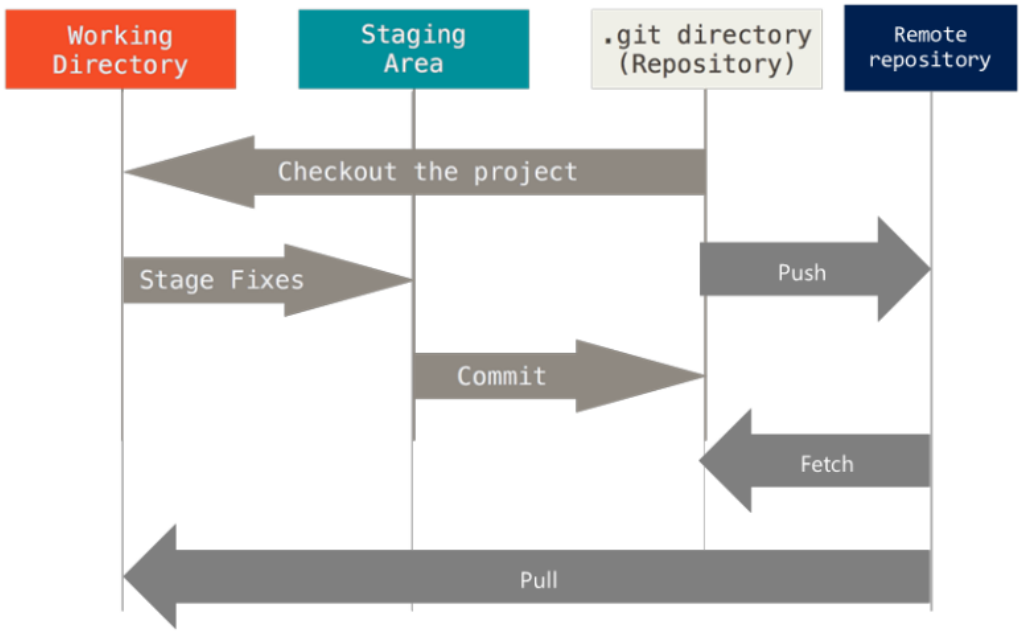
Commit한 상태 Staging Area-> Repository 로 이동

```
jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial/git-repo (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 217 bytes | 217.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/jojun01835/repo-test.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Push를 사용하여 원격 저장소에 파일 올리기



원격 저장소에 올라간 파일



깃 명령어 활용

$ git init

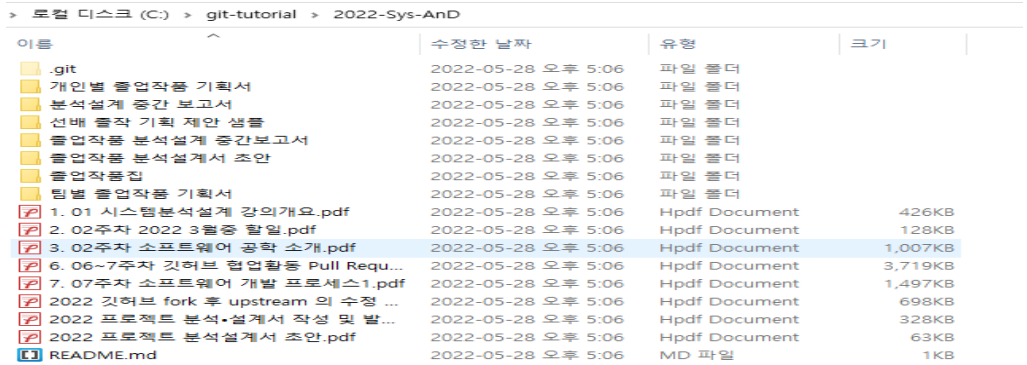
- 현재 디렉토리에 git을 사용할 수 있도록 초기화

```
jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial
$ git init
Initialized empty Git repository in C:/git-tutorial/.git/
```

$ git clone

- 깃 클론 명령어를 치고 URL를 입력하면 깃허브에 있는 파일들을 모두 복사

```
jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial
$ git clone https://github.com/ai7dnn/2022-Sys-AnD.git
```



$ git config --global

- 깃의 환경설정 -global을 입력하면 전역설정으로 바뀜

```
jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial/git-repo (main)
$ git config --global user.name jojun01835
```

$ git add

- 생성한 파일을 Working Dir에 가져온다 Untrack파일이 track영역에 들어온다.

```
jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial/hello (main)
$ git add.
git: 'add.' is not a git command. See 'git --help'.

The most similar command is
        add
```

$ git commit -m " "

- Add한 파일을 repository에 가져온다 -m 뒤에는 코멘트를 달아서 무엇이 최신화 되어있는지를 보여준다

```
jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial/hello (main)
$ git commit -m "add file"
[main (root-commit) 28bb21d] add file
 1 file changed, 1 insertion(+)
 create mode 100644 hello.py
```

$ git log , git log --oneline

- 깃의 커밋내역을 보여준다 -oneline를 사용하면 한 줄로 보여준다

```
$ git log
commit 28bb21d3c03ced2d6b2a7bac1e3faa6d0671a81d (HEAD -> main)
Author: jojun01835 <jojun01835@gmail.com>
Date:   Sat May 28 17:33:23 2022 +0900

    add file

jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial/hello (main)
$ git log --oneline
28bb21d (HEAD -> main) add file
```

깃 명령어 활용

$ git status
- 파일의 상태를 보여준다

```
jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial/hello (main)
$ echo "printf("hello c")" >> hello.c

jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial/hello (main)
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hello.c

nothing added to commit but untracked files present (use "git add" to track)
```

$ git remote add (원격 저장소명) [원격저장소URL] , remote -v
- 원격저장소와 로컬 저장소를 연결한다
- Remote -v 는 현재 연결된 저장소를 보여준다

```
jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial/git-hello (master)
$ git remote add origin https://github.com/jojun01835/repo01.git

jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial/git-hello (master)
$ git remote -v
origin  https://github.com/jojun01835/repo01.git (fetch)
origin  https://github.com/jojun01835/repo01.git (push)
```

$ git push -u (저장소 이름) (브랜치 이름)
- 연결된 원격 저장소로 파일을 옮긴다

```
jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial/git-hello (master)
$ git push -u origin master
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 12 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (12/12), 1.42 KiB | 1.42 MiB/s, done.
Total 12 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/jojun01835/repo01/pull/new/master
remote:
To https://github.com/jojun01835/repo01.git
```

$ git checkout 명령어
- Git checkout head -명령어는 깃의 커밋상태를 되돌림
- Head -- 를 사용하면 두단계 전으로 이동

```
jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial/git-head (main)
$ git log --oneline
a5bc11b (HEAD -> main) 3rd commit
d010c67 2nd commit
4dfe584 first commit
```

```
jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial/git-head ((d010c67...))
$ git log --oneline
d010c67 (HEAD) 2nd commit
4dfe584 first commit
```

```
jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial/git-head ((4dfe584...))
$ git log --oneline
4dfe584 (HEAD) first commit
```

$ git checkout 명령어
- Git checkout (브랜치 이름)
- 가장 최신의 커밋으로 이동한다

```
jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial/git-head ((4dfe584...))
$ git checkout main
Previous HEAD position was 4dfe584 first commit
Switched to branch 'main'

jojun@DESKTOP-18Q3E6Q MINGW64 /c/git-tutorial/git-head (main)
$ git log --oneline
a5bc11b (HEAD -> main) 3rd commit
d010c67 2nd commit
4dfe584 first commit
```

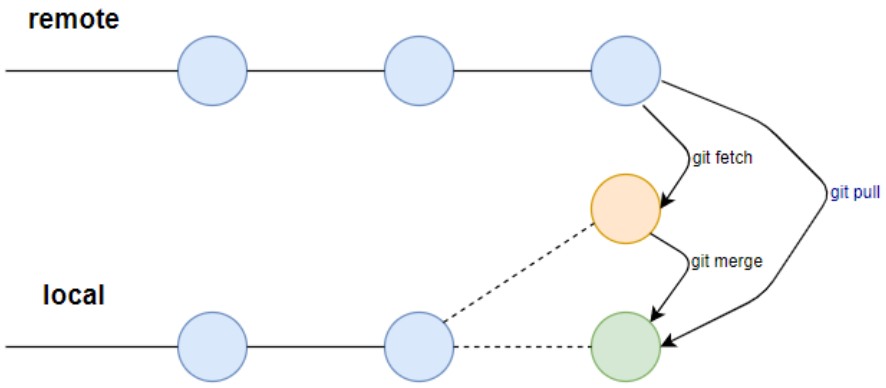
Fetch 와 Merge

Git pull 명령
- Fetch 명령과 병합하는 merge 명령이 순차적으로 진행

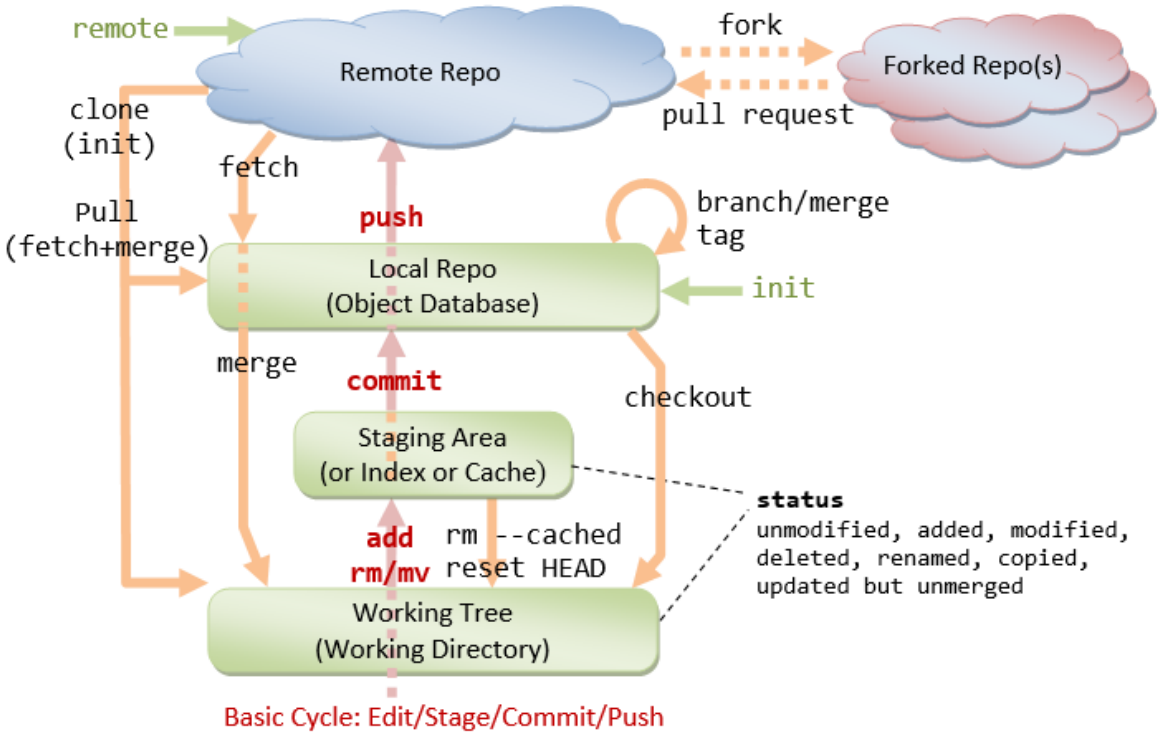
Fetch 명령
- 원격 저장소의 정보를 로컬 저장소로 가져오는 명령

Merge 명령
- 변경된 정보를 로컬 저장소의 내용과 병합

Fetch 명령으로 작업한 내용을 확인한 후
- Merge 여부를 결정하는 과정
  - 병합(merge)
    - 다른 브랜치의 내용을 합침



Git life cycle



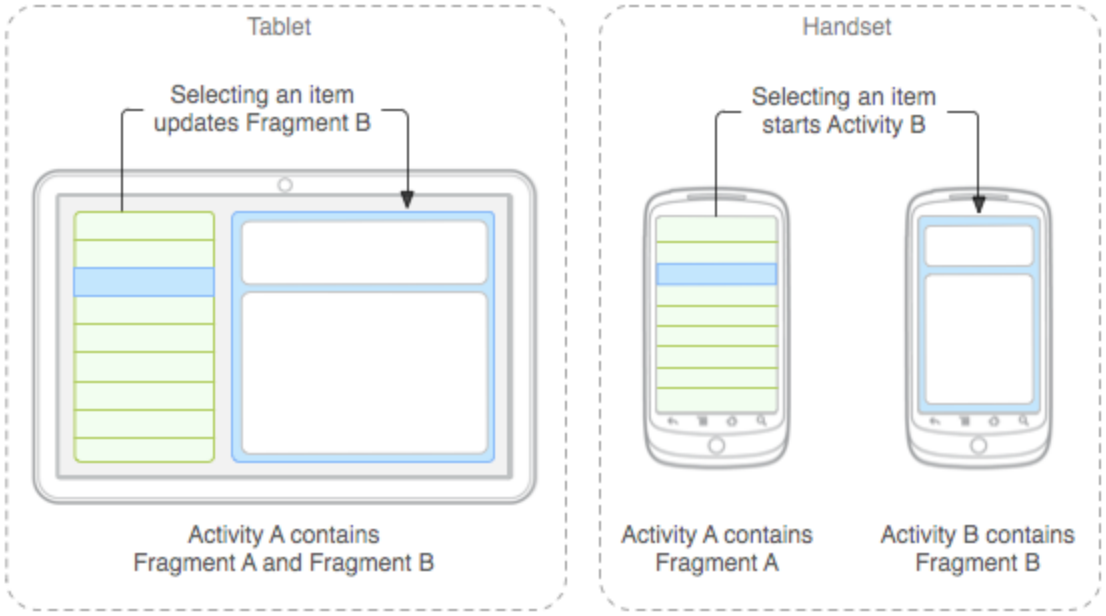
Chapter 2.

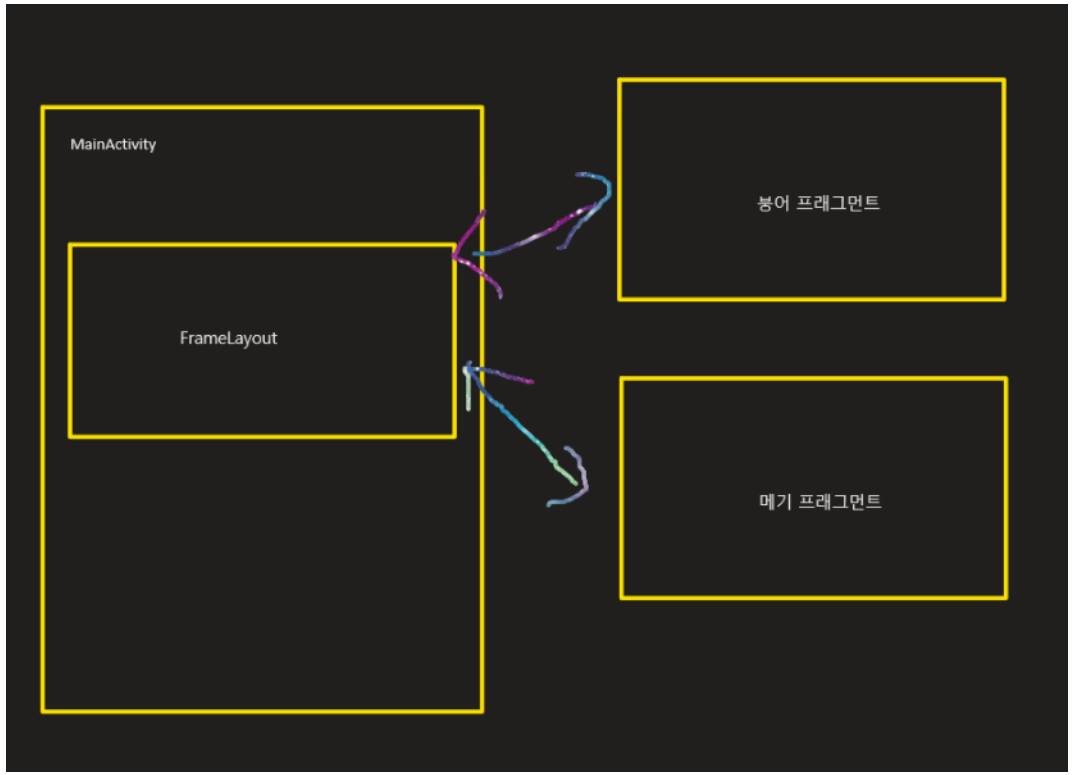
안드로이드

안드로이드 Fragment

Fragment 개요

- 프래그먼트는 액티비티 내에서 화면 UI의 일부를 나타냅니다.
- 여러 개의 프래그먼트를 조합하여 액티비티가 출력하는 한 화면의 UI를 표현할 수 있습니다
- 하나의 프래그먼트를 다른 액티비티에 재사용할 수 있습니다.





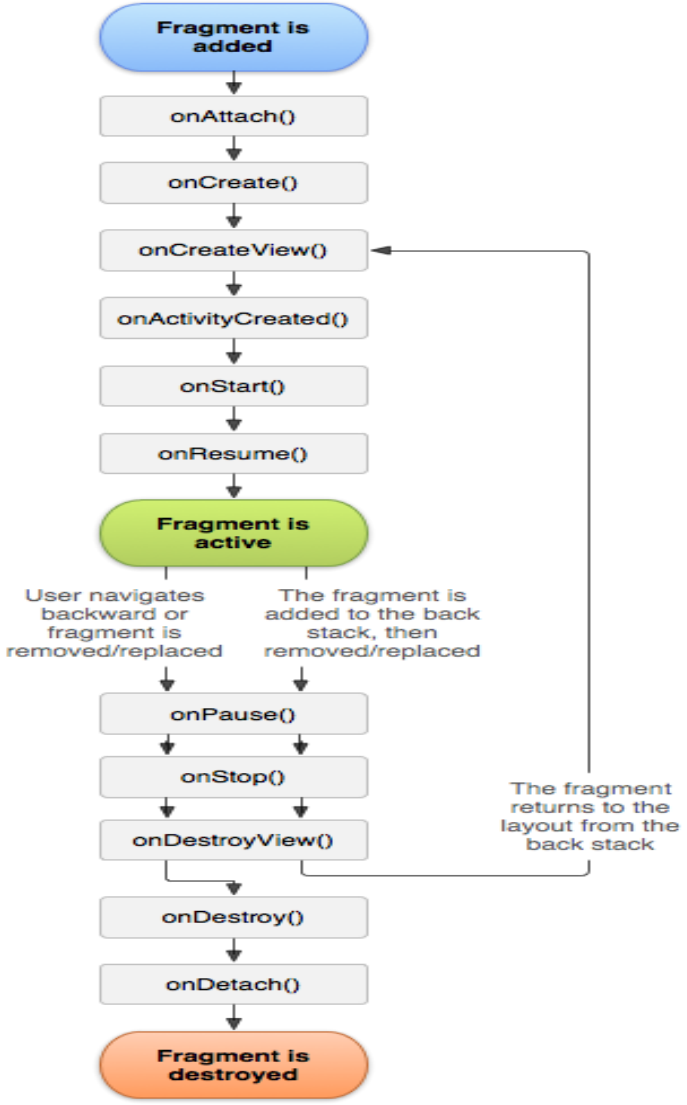
위와 같은 그림으로 MainActivity에

프래그먼가 들어갈 틀을 선언하고,

틀 위에 올려놓을 프래그먼트를 여러 개

만들어 끼우고 빼고 하는게 가능하다.

안드로이드 Fragment



Fragment LifeCycle

1. **onCreate()**

프래그먼트를 생성할 때 호출합니다. 프래그먼트가 일시정지 혹은 중단 후 재개되었을 때 유지하고 있어야 하는 것을 여기서 초기화 합니다.

2. **onCreateView()**

프래그먼트가 자신의 인터페이스를 처음 그리기 위해 호출합니다. View를 반환해야 합니다. 이 메서드는 프래그먼트의 레이아웃 루트이기 때문에 UI를 제공하지 않는 경우에는 null을 반환합니다

3. **onPause()**

사용자가 프래그먼트를 떠나면 첫번 째로 이 메서드를 호출합니다. 사용자가 돌아오지 않을 수도 있으므로 여기에 현재 사용자 세션을 넘어 지속되어야 하는 변경사항을 저장합니다.

안드로이드 Fragment 구현절차

1. activity_main.xml 작성

```
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/fragmentFrame"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"/>

<LinearLayout
    android:id="@+id/linearLayout"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:gravity="bottom"
    android:orientation="horizontal"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent" />

<Button
    android:id="@+id/buttonB"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="5dp"
    android:layout_weight="1"
    android:text="봉어 프래그먼트"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="1.0" />

<Button
    android:id="@+id/buttonM"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="5dp"
    android:layout_weight="1"
    android:text="메기 프래그먼트"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.555"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.907" />
```

2. 화면이동을 보여 줄 두개의 xml파일 작성

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:and
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/purple_500">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="메기 Fragment"
        android:textSize="32sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:androi
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/teal_200">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="봉어 Fragment"
        android:textSize="32sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

안드로이드 Fragment 구현절차

3.두개의 java.class 파일을 만든다.

```
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

public class FragmentM extends Fragment {
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle saveInstanceState){
        return inflater.inflate(R.layout.fragment_m,container,false);
    };
}
```

```
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

public class FragmentB extends Fragment {
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle saveInstanceState){
        return inflater.inflate(R.layout.fragment_b,container,false);
    };
}
```

inflate 함수

```
/**
     * Inflate a new view hierarchy from the specified xml resource. Throws
     * {@link InflateException} if there is an error.
     *
     * @param resource ID for an XML layout resource to load (e.g.,
     *        <code>R.layout.main_page</code>)
     * @param root Optional view to be the parent of the generated hierarchy (if
     *        <em>attachToRoot</em> is true), or else simply an object that
     *        provides a set of LayoutParams values for root of the returned
     *        hierarchy (if <em>attachToRoot</em> is false.)
     * @param attachToRoot Whether the inflated hierarchy should be attached to
     *        the root parameter? If false, root is only used to create the
     *        correct subclass of LayoutParams for the root view in the XML.
     * @return The root View of the inflated hierarchy. If root was supplied and
     *         attachToRoot is true, this is root; otherwise it is the root of
     *         the inflated XML file.
     */
    public View inflate(@LayoutRes int resource, @Nullable ViewGroup root, boolean
attachToRoot)
```

resource : 객체화 될 프래그먼트 xml을 의미합니다

root : 객체화 될 View의 parent layout을 지정해 주며 선택사항입니다. attachToRoot가 true이면 parent에 객체화 된 View를 바로 add 해 버리고, false일 경우 parent 레이아웃의 레이아웃 파라미터만 받아옵니다

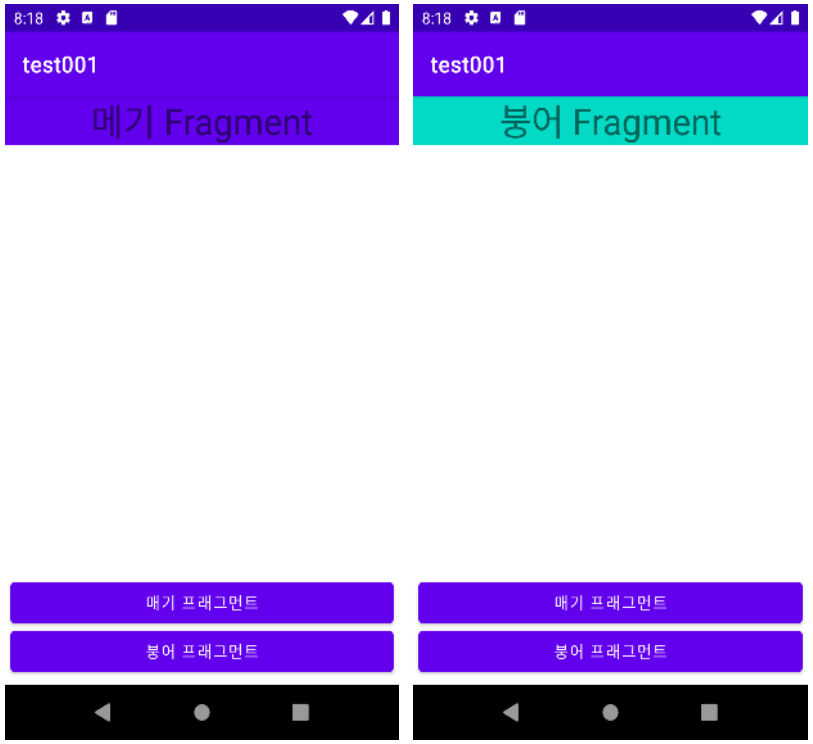
attachToRoot : 부모 ViewGroup에 프래그먼트가 바로 자식으로 들어갈지 말지를 결정합니다

안드로이드 Fragment 구현절차

4.MainActivity 파일을 작성한다.

```java
public class MainActivity extends AppCompatActivity{
    private FragmentB fragmentB;
    private FragmentM fragmentM;
    private Button buttonB;
    private Button buttonM;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        fragmentB = new FragmentB();
        fragmentM = new FragmentM();

        //프래그먼트 매니저 획득
        FragmentManager fragmentManager = getSupportFragmentManager();
        //프래그먼트를 올리거나 교체하는 작업은 Transaction 입니다.
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
        //프래그먼트를 FrameLayout의 자식으로 등록합니다.
        fragmentTransaction.add(R.id.fragmentFrame,fragmentB);
        //commit을 하면 자식으로 등록된 프래그먼트가 화면에 보여집니다.
        fragmentTransaction.commit();

        buttonB = findViewById(R.id.buttonB);
        buttonM = findViewById(R.id.buttonM);

        buttonB.setOnClickListener(v -> {
            FragmentManager fm1 = getSupportFragmentManager();
            FragmentTransaction ft1 = fragmentManager.beginTransaction();
            ft1.replace(R.id.fragmentFrame, fragmentB);
            ft1.commit();
        });
        buttonM.setOnClickListener(v -> {
            FragmentManager fm2 = getSupportFragmentManager();
            FragmentTransaction ft2 = fragmentManager.beginTransaction();
            ft2.replace(R.id.fragmentFrame, fragmentM);
            ft2.commit();
```

결과 화면



위의 결과화면 같이 두개의 xml파일을 번갈아 가면서 보여준다.

안드로이드 크롤링

Jsop

- Jsoup란 특정 url의 html을 파싱해주는 라이브러리

- 파싱은 어떤 페이지(문서, html 등)에서 내가 원하는 데이터를 특정 패턴이나 순서로 추출해 가공하는 것을 말한다

안드로이드 크롤링 구현절차

1. manifest파일과 build.gradle 수정

```
implementation 'org.jsoup:jsoup:1.11.3'
```

app 모듈의 build.gradle에 jsoup 추가

```
<uses-permission android:name="android.permission.INTERNET" />
```

Manifest파일에 권한 추가

2. xml파일 수정

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://sc
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/number"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

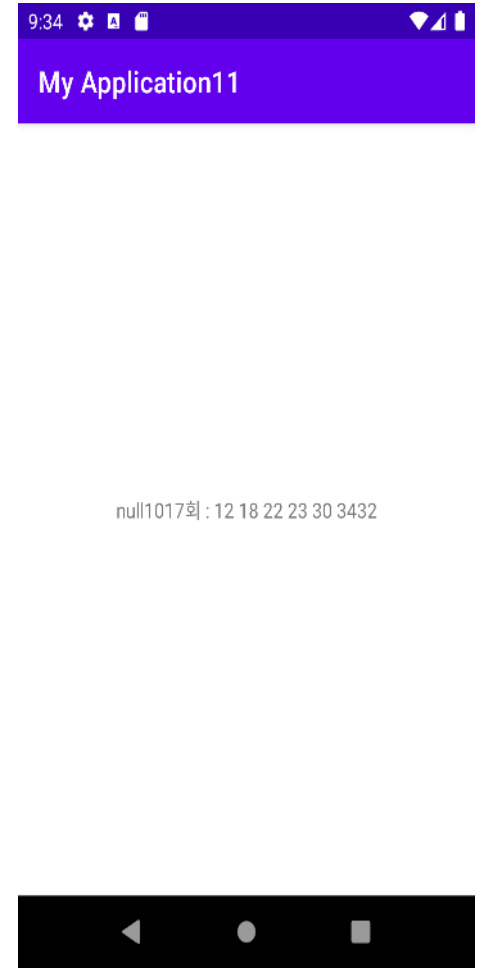
안드로이드 크롤링

2. 크롤링할 사이트 찾고 원하는 데이터의 위치를 찾아보기


```
<h4>…</h4>
<p class="desc">(2022년 05월 28일 추첨)</p>
<div class="nums">
<div class="num win"> == $0
<strong>당첨번호</strong>
<p>
<span class="ball_645 lrg ball2">12</span>
<span class="ball_645 lrg ball2">18</span>
<span class="ball_645 lrg ball3">22</span>
<span class="ball_645 lrg ball3">23</span>
<span class="ball_645 lrg ball3">30</span>
<span class="ball_645 lrg ball4">34</span>
::after
</p>
</div>
<div class="num bonus">…</div>
::after
</div>
```

3. MainActivity작성 , 결과확인

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    textView = (TextView) findViewById(R.id.number);
    final Bundle bundle = new Bundle();
    new Thread() {
        public void run() {
            Document doc = null;
            try {
                doc = Jsoup.connect("https://dhlottery.co.kr/common.do?method=main").get();
                Elements contents = doc.select("#LottoDrwNo");
                nums += contents.text() + "회 :";

                for (int i = 1; i < 7; i++){
                    contents = doc.select("#drwtNo"+i);
                    nums += " "+ contents.text();
                }
                nums += doc.select("#bnusNo").text();

                bundle.putString("numbers",nums);
                Message msg = handler.obtainMessage();
                msg.setData(bundle);
                handler.sendMessage(msg);

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }.start();
}
Handler handler = new Handler(){
    @Override
    public void handleMessage(Message msg){
        Bundle bundle = msg.getData();
        textView.setText(bundle.getString("numbers"));
    }
```



Chapter 3.

아두이노

아두이노 라이브러리

DHT11 온습도 센서 주요 함수

- DHT(uint8_t pin, uint8_t type, uint8_t count = 6)
  - Pin : DHT 연결 핀 번호
  - Type :센서 Type으로 DHT11, DHT12, DHT22, DHT21,AM2301 중의 하나
  - Count : 센서의 개수
- DHT.begin() : DHT 사용 선언
- Float.DHT.readTemperature(bool S = fasle, bool force = false) : 화씨, 섭씨 중 선택된 온도 단위 반환
  - S : ture 입력 시 화씨, false 입력 시 섭씨
- Float DHT.converCtoF(float) : 섭씨를 화씨로 반환
- Float DHT.converFtoC(float) : 화씨를 섭씨로 변환
- Float computeHeatindex(float temperature, float percentHumidity, bool isFahrenheit = ture
  - Temperature : 온도값
  - percentHumidity : 습도값
  - IsFahrenheit : true 입력 시 화씨, false 입력 시 섭씨
- Float readHumidity(bool force = false) : 습도를 %단위로 반환

```
DHT::DHT(uint8_t pin, uint8_t type, uint8_t count) {
  (void)count; // Workaround to avoid compiler warning.
  _pin = pin;
  _type = type;
#ifdef __AVR
  _bit = digitalPinToBitMask(pin);
  _port = digitalPinToPort(pin);
#endif
  _maxcycles =
      microsecondsToClockCycles(1000); // 1 millisecond timeout for
                                       // reading pulses from DHT sensor.
  // Note that count is now ignored as the DHT reading algorithm adjusts itself
  // based on the speed of the processor.
}
```

```
void DHT::begin(uint8_t usec) {
  // set up the pins!
  pinMode(_pin, INPUT_PULLUP);
```

```
float DHT::computeHeatIndex(float temperature, float percentHumidity,
                            bool isFahrenheit) {
  float hi;

  if (!isFahrenheit)
    temperature = convertCtoF(temperature);

  hi = 0.5 * (temperature + 61.0 + ((temperature - 68.0) * 1.2) +
              (percentHumidity * 0.094));

  if (hi > 79) {
    hi = -42.379 + 2.04901523 * temperature + 10.14333127 * percentHumidity +
         -0.22475541 * temperature * percentHumidity +
         -0.00683783 * pow(temperature, 2) +
         -0.05481717 * pow(percentHumidity, 2) +
         0.00122874 * pow(temperature, 2) * percentHumidity +
         0.00085282 * temperature * pow(percentHumidity, 2) +
         -0.00000199 * pow(temperature, 2) * pow(percentHumidity, 2);

    if ((percentHumidity < 13) && (temperature >= 80.0) &&
        (temperature <= 112.0))
      hi -= ((13.0 - percentHumidity) * 0.25) *
            sqrt((17.0 - abs(temperature - 95.0)) * 0.05882);

    else if ((percentHumidity > 85.0) && (temperature >= 80.0) &&
             (temperature <= 87.0))
      hi += ((percentHumidity - 85.0) * 0.1) * ((87.0 - temperature) * 0.2);
  }

  return isFahrenheit ? hi : convertFtoC(hi);
}
```

```
float DHT::readTemperature(bool S, bool force) {
  float f = NAN;

  if (read(force)) {
    switch (_type) {
    case DHT11:
      f = data[2];
      if (data[3] & 0x80) {
        f = -1 - f;
      }
      f += (data[3] & 0x0f) * 0.1;
      if (S) {
        f = convertCtoF(f);
      }
      break;
    case DHT12:
      f = data[2];
      f += (data[3] & 0x0f) * 0.1;
      if (data[2] & 0x80) {
        f *= -1;
      }
      if (S) {
        f = convertCtoF(f);
      }
      break;
    case DHT22:
    case DHT21:
      f = ((word)(data[2] & 0x7F)) << 8 | data[3];
      f *= 0.1;
      if (data[2] & 0x80) {
        f *= -1;
      }
      if (S) {
        f = convertCtoF(f);
      }
      break;
```

아두이노 라이브러리

Servo 라이브러리 주요함수

- Attach() 함수
  - Servo.attach(pin,min,max)
    - Servo : Servo 함수 객체명
    - Pin : 서보 모터 연결 핀 Arduino Uno 보드의 경우 9번 또는 10번 핀
    - Min : 최소 각도(0°)일 때 HIGH 유지 시간
    - max : 최대 각도(180°)일 때 HIGH 유지 시간
- detach() 함수
  - servo.detach() 와 같이 사용 가능하며 서보 모터에 할당된 핀 사용을 중지합니다. servo.datach() 후에 9번핀 10번핀을 PWM 포트로 사용 가능합니다.
- write()
  - servo.write(angle) 와 같이 사용 가능하며, angle 에 입력된 0° ~ 180° 범위 내 값으로 서보 모터를 제어합니다.
- writeMicroseconds()
  - servo.writeMicroseconds(microSeconds) 와 같이 사용가능하며 microSeoncds 에 입력된 us 단위 시간 만큼 HIGH Pulse 상태를 유지합니다. 일반적으로 사용되는 Servo Motor 가 아닌 경우 시간을 직접 입력하여 서보 모터를 제어할 수 있습니다.

  - servo.read() 서보 모터에 마지막으로 설정한 각도 값을 반환합니다. Servo 모터 현재 상태를 읽어 오는 것이 아니라, 마지막으로 servo.write() 함수로 설정 한 한 값을 기억했다가 반환해줍니다.

```
public:
  Servo();
  uint8_t attach(int pin);           // attach the given pin to the
  uint8_t attach(int pin, int min, int max); // as above but also s
  void detach();
  void write(int value);             // if value is < 200 its treat
  void writeMicroseconds(int value); // Write pulse width in micros
  int read();                        // returns current pulse width
  int readMicroseconds();            // returns current pulse width
  bool attached();                   // return true if this servo i
private:
   uint8_t servoIndex;               // index into the channel data
   int8_t min;                       // minimum is this value times
   int8_t max;                       // maximum is this value times
};

#endif
#endif
```

Chapter 4.

소프트웨어 공학

소프트웨어 공학

소프트웨어 공학 개요

- 소프트웨어 공학은 소프트웨어의 개발, 운용, 유지보수 등의 생명 주기 전반을 체계적이고 서술적이며 정량적으로 다루는 학문이다 즉, 공학을 소프트웨어에 적용하는 것이다.
- 소프트웨어 공학이라는 용어가 처음 나타난 곳은 1968년 나토 소프트웨어 공학 학회로, 당시에는 소프트웨어 위기에 관해 사람들이 주의를 기울여 생각할 것을 장려하기 위해서였다.
- 그 이후로, 하나의 직업으로서, 또한 학문의 한분야로서 꾸준히 품질, 비용, 유지 보수성 빌드 속도가 개선된 소프트웨어를 창조하는데 전념해 왔다.
- 이 분야는 그 자매 분야인 공학에 비해 아직도 상대적으로 젊은 분야로, 소프트웨어공학'이란 실제로 무엇이며 전통적인 공학의 정의에 부합하는지에 대한 논의가 이루어지고 있다
- 소프트웨어를 단순히 프로그래밍으로만 보는 한계를 벗어나는 것으로부터 유기적으로 성장한분야이다. 최근의 흐름으로는 관점 지향(Aspect), 애자일(Agile), 모델주도(Model-Driven) 등이 있다.

소프트웨어위기

- 소프트웨어 위기란 소프트웨어가 더 이상 하드웨어의 개발 속도를 따라가지 못하거나
- 소프트웨어가 더 이상 사용자의 요구를 충족시킬 수가 없음

극복방안

- 소프트웨어 설계 및 개발에 대한 전문적인 연구 및 방법론 도출 필요
- 자동화 : CASE,MUL/ERD,형상관리
- 품질보증 : ITIL, CMMi, SPICE
- 공학적 접근 : 구조적 방법론, 정보공학 방법론, 객체지향 방법론 , CBD 방법론
- 표준화 : ISO, IEC, IEEE, IEFT, W3C

소프트웨어 개발 방법론

소프트웨어 개발 방법론

- 소프트웨어 개발방법론은 소프트웨어를 어떻게 만들지에 대해 관심을 가진다. 따라서 개발 방법론에는 단계별 산출물뿐만 아니라 산출물은 누가 어떤 순서로 어떻게 만들어야 하는지 그리고 어떤 도구를 사용해야 하는지 구체적으로 정의한다.

역사

•1969년까지 구조적 프로그래밍이 주로 쓰였다.

•1980년대 구조적 시스템 분석과 설계 방법론이 쓰였다.

•1990년대 객체 지향 프로그래밍이 1960년대부터 개발되어, 1990년대 중반에 주류 개발 방법론이 된다.

•고속 개발 방법론이 1991년부터 쓰인다.

•스크럼이 1990년 후반부터 쓰인다.

•SEI의 와츠 험프리가 팀 소프트웨어 프로세스를 개발한다.

•2000년대 익스트림 프로그래밍이 1999년부터 쓰인다.

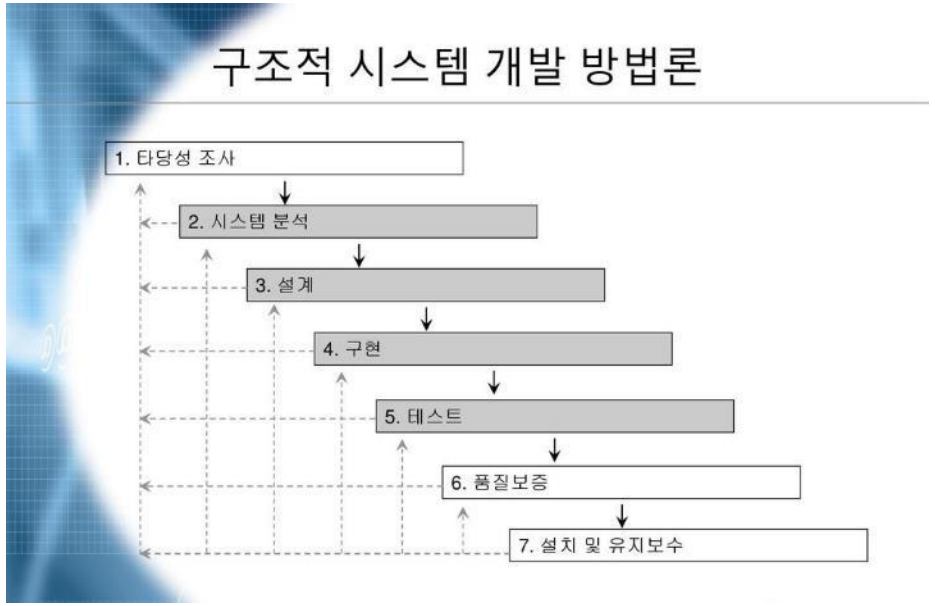
•래셔널 통합 프로세스 (RUP)가 1998년부터 쓰인다.

•스콧 앰블러가 2005년에 애자일 통합 프로세스 (AUP)를 시작한다.

소프트웨어 개발 방법론

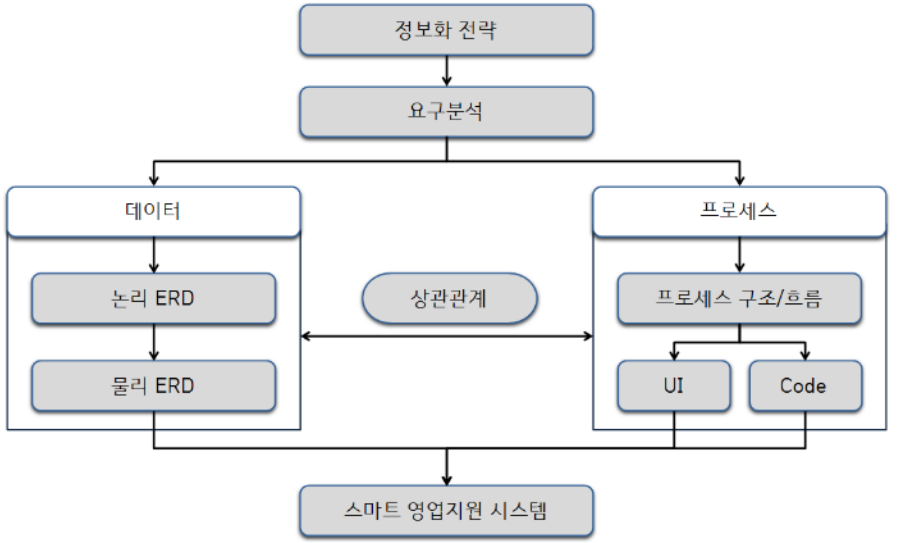
소프트웨어 개발 방법론

- 구조적 방법론
  - 구조적 방법론은 정형화된 분석 절차에 따라 사용자 요구사항을 파악하여 문서화하는 처리중심의 방법론
  - 1960년대까지 가장 많이 적용되었던 소프트웨어 개발 방법론
  - 쉬운 이해 및 검증이 가능한 프로그램 코드를 생성하는 것이 목적이다
  - 복잡한 문제를 다루기 위해 분할과 정복 원리를 적용한다



소프트웨어 개발 방법론

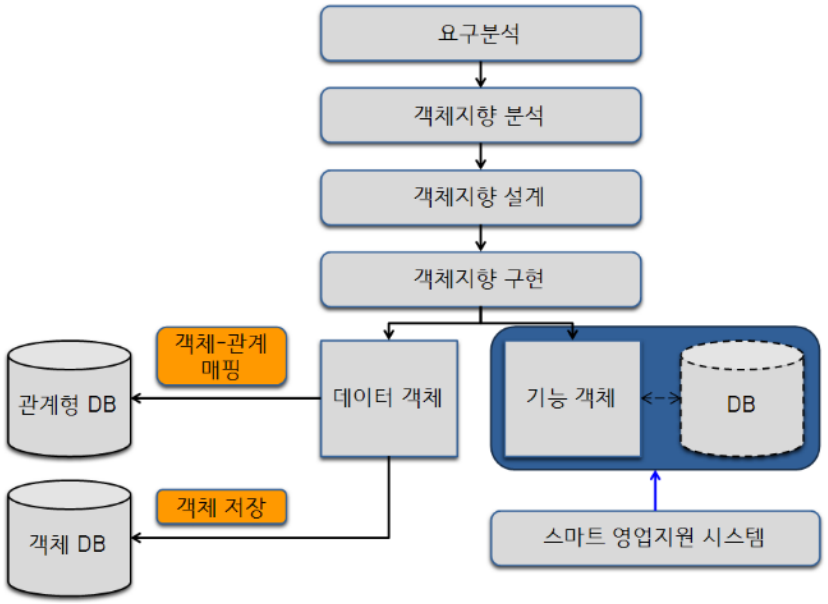
- 정보공학 방법론
  - 정보공학 방법론은 정보 시스템의 개발을 위해 계획, 분석, 설계 구축에 정형화된 기법들을 상호 연관성 있게 통합 및 적용하는 자료 중심의 방법론
    - 설계할 때는 스토리보드, 데이터 설계서, UI 설계서 정도를 작성하고 개발 과정에서는 이를 바탕으로 물리적 테이블과 프로그램 코드가 만들어 진다.



소프트웨어 개발 방법론

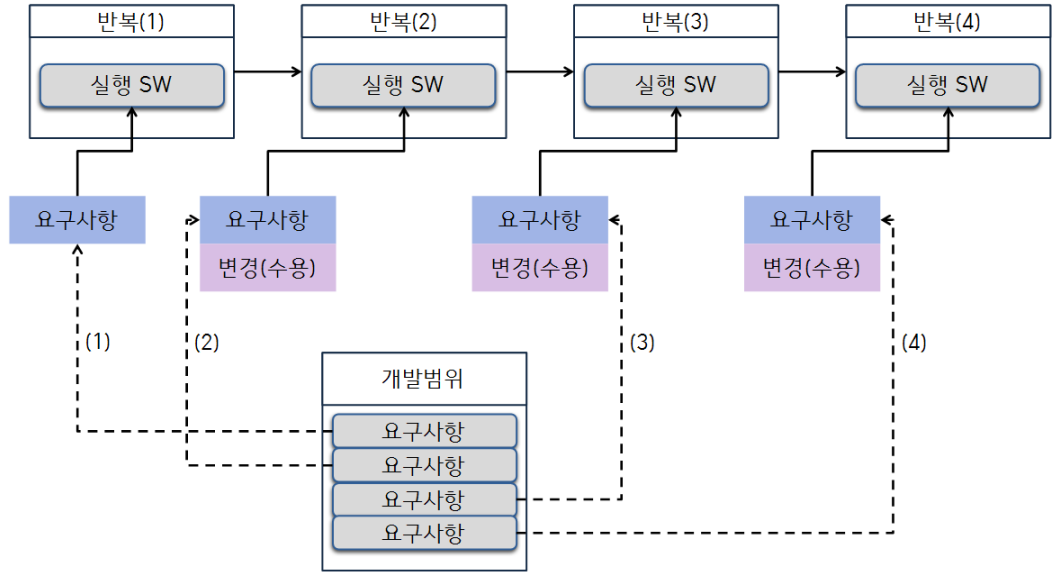
소프트웨어 개발 방법론

- 객체지향 개발 방법론
  - 객체지향 방법론은 현실 세계의 개체를 기계의 부품처럼 하나의 객체로 만들어 소프트웨어를 개발할 때 기계의 부품을 조립하듯이 객체들을 조립해서 필요한 소프트웨어를 구현하는 방법론
  - 객체지향 방법론은 구조적 기법의 문제점으로 인한 소프트웨어 위기의 해결책으로 채택되었다.



소프트웨어 개발 방법론

- 애자일 방법론
  - 애자일은 민첩한, 기민한 이라는 의미로 애자일 방법론은 고객의 요구사항 변화에 유연하게 대응할 수 있도록 일정한 주기를 반복하면서 개발 과정을 진행하는 방법론이다.
  - 소규모 프로젝트, 고도로 숙달된 개발자, 급변하는 요구사항에 적합하다
  - 애자일 방법론의 대표적인 종류에는 XP, 스크럼, 칸반, 크리스탈 등이 있다



참조한 사이트

깃허브 https://github.com/jojun01835/github-basic

깃허브 https://backlog.com/gittutorial/kr/stepup/stepup3_2.html

안드로이드 https://developer.android.com/guide/components/fragments?hl=ko

안드로이드 https://tosuccess.tistory.com/119

아두이노 https://juahnpop.tistory.com/156

아두이노 https://juahnpop.tistory.com/123

소프트웨어 공학

https://ko.wikipedia.org/wiki/%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4_%EA%B3%B5%ED%95%99

개발방법론

http://wiki.hash.kr/index.php/%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4_%EA%B0%9C%EB%B0%9C%EB%B0%A9%EB%B2%95%EB%A1%A0

[ ]

감사합니다

20210996 조준영