**3/15 – 11 장 java.lang.Object 클래스**

**12 장 멀티스레드**

**3/16,17 – 자바 x**

**git 특상**

**3/18 – 15장 컬렉션 프레임워크**

**13장 제네릭**

**3/19 - 14장 람다식**

**3/ 22 - 18장 입출력**

**3/23 - 18장 네트워크**

**(사이트 url – 과제1개, 시험)**

**- java.lang.Object**

**1>자바 모든 클래스 자동 상속**

**2> import 자동**

**3> 메소드 다른 클래스 자동 포함**

**4> 그대로 사용 또는 overriding**

**5> toString- 객체 출력 문장 자동 호출**

**overriding 하면 객체 출력 내용 결정**

**6> equals – 객체 (주소값 )동등성 비교**

**overriding 하면 객체 주소가 아닌 다른 부분 동등성 비교**

**결정**

**java.lang.30 여개 크랫스 > 메소드 다수개**

**- java.lang.String p496**

**1> 문자열 저장 객체**

**2> String만 가지는 특징**

**2-1. new 키워드 없이 문자열 표현 가능**

**String s1 = new String("java");**

**String s1 = "java";**

**int i = 10;**

**2-2. + 연산자 사용 문자열 결합**

**( + - 숫자 덧셈 / 문자열 결합)**

**s.o.p("100" + 200)--> "100200"(String)**

**s.o.p("100" + "200")--> "100200"(String)**

**s.o.p(100 + 200)--> 300(int)**

**- String**

**특징**

**생성자 오버로딩 – 문자열 데이터 사용 방법 다양하게 제공**

**"자바"**

**"JAVA"**

**char c[] = { 'a', 'b', 'c', 'd' }===> "abcd"**

**String sc = new String(c);===>"abcd"**

**String sc = new String(c, 1, 2);==>"bc"**

**String sc = new String(c, 1);==>"bcd"**

**byte b [] = {65, 66, 67, 68}===>"ABCD"**

**String sb = new String (b);**

**(byte 1 바이트(8비트) 정수- : -2^7 ~ 2^7-1 : -128 ~ 127 )**

**'0' -> 48 'A'--> 65( 정수 ) 'a'-->97**

**메소드 - 50여개 – 문자열 표현 조작**

**String s = new String("java");**

**s.length();-->4 문자(문자 1개 2바이트 표현 - 8바이트)**

**배열변수명.length**

**s.toUpperCase() / s.toLowerCase()**

**s.replace('a', 'b') / s.replace("ja", "ka")**

**s.equals() / s1.equalgnoreCase()**

**String s1 = "java";**

**String s2 = new String("java");**

**String s3 = "JAVA";**

**s1 == s2 --> 주소값 - fasle**

**s1.equals(s2) --> 문자열값(대소문자 구분) – true**

**s1.equalgnoreCase(s3) --> 문자열값(대소문자 미구분) – true**

**(**

**class String extends Object {**

**public boolean equals(Object o){**

**오버라이딩 – 문자열값** 비교

**}**

**}**

**String s = "program";**

**s.indexOf("ram"); -->4**

**s.indexOf("Lam"); -->-1**

**if( s.indexOf("Lam") >= 0) {  
}**

**s.substring(4, 7);//ram**

**s.substring(4);//ram**

**"100" --> 100**

**Integer.parseInt("100")**

**100 ---> "100"**

**String.valueOf(100)**

**String s = "program";**

**'p'<===s.charAt(0)**

**String s = "100 김사원 34500.5 사원 총무부";**

**String[] result = s.split(" ");**

**e.id = result[0];**

**e.name = result[1];**

**....**

**String s = "program";**

**s.o.p(s + " 과정" );**

**===> program 과정**

**s.o.p(String.format("%20s", s));**

**==> 20자리 문자 형태**

**program 과정**

**String – 기본 문자열 표현 조작 50여개 기능**

**String s = "java";==>4 개 문자열 저장 생성-수정 불가능**

**s = s + "program";==> 문자열 추가 변경 많으면 메모리 낭비**

**StringBuffer sb = new StringBuffer("java");**

**sb.append("program");**

|  |  |
| --- | --- |
| **sb:300**  **s:200** | **100:[java]-참조변수없다**  **(메모리 삭제 가능)**  **200:[javaprgram]**  **300:[javaprogram]** |

**StringBuilder / StringBuffer – 문자열 생성 이후에 문자열 변경 추가 작업 전용 – "+"**

**StringTokenizer – 토큰 분리 전용- split()**

**====================================**

**- java.util.regext.Pattern – 517p**

**정규표현식 – 문자열 정해진 형태 구성 필요. 정규형태 맞는 문자열인가 검사식.**

**- aaa????bbb**

**- 010-국번3-4자리번호-4자리번호**

**- 이메일 문자나 숫자@문자나 숫자**

**-** [**http://www.oracle.com**](http://www.oracle.com)

[**http://zoom.us**](http://zoom.us)

**http://** [**www.oracle.co.kr**](http://www.oracle.co.kr)

|  |  |
| --- | --- |
| **[abc]** | **문자 1개가 a나 b나 c만 가능** |
| **[a-zA-Z0-9]** | **문자 1개가 알파벳 소,대문자,숫자중 1개 가능** |
| **\w** | **[a-zA-Z0-9]** |
| **[0-9]** | **문자 1개가 숫자 구성** |
| **\d** | **[0-9]** |
| **[abc]{3}** | **a나 b나 c만 구성된 문자열 3개** |
| **[abc]{3,5}** | **a나 b나 c만 구성된 문자열 3-5개** |
| **[abc]{3,}** | **a나 b나 c만 구성된 문자열 3개이상** |
| **[abc]+** | **a나 b나 c만 구성된 문자열 1개이상** |
| **[abc]\*** | **a나 b나 c만 구성된 문자열 0개이상** |
| **[abc]?** | **a나 b나 c만 구성된 문자열 0-1개** |

**[표현형태]{횟수}**

**538p- 오후 1시부터**

**- 날짜와 시간 표현**

**- java.util.Date/Calendar/java.text.SimpleDateFormat**

**jdk 8 – 170 여개 패키지**

**여러 클래스들 = 중복 기능**

**jdk 업그레이드 – 라이브러리 추가**

**기본형/참조형( 배열 enum class interface)**

**java.util.Date**

**Date now = new Date();===> 현재시스템날짜시간**

**Date now = new Date(1/1000 초);===>**

**1970년 1월 1일 0시 0분 0초 ~ 경과한 날짜시각**

**Date now = new Date(2021, 3, 15);===>**

**Date now = new Date(2021, 3, 15, 12, 12, 12);===>**

**==> 사용 자제 권고**

**Date 대부분 메소드 사용 자제 권고**

**===> 대부분 기능 Calendar 클래스 이용 권고**

**- Calendar cal = new Calendar();**

**-->error(abstract class + private 생성자)**

**Calendar cal = Calendar.getInstance();**

**==> singleton pattern**

**( private 생성자,**

**public static Calendar getInstance(){**

**if( cal == null){**

**c = new Calendar();**

**}**

**return c;**

**}**

**cal.get(Calendar.YEAR);**

**cal.get(Calendar.MONTH) + 1;**

**cal.get(Calendar.DAY\_OF\_MONTH);**

**cal.get(Calendar.HOUR);**

**cal.get(Calendar.MINUTE);**

**cal.get(Calendar.SECOND);**

**java.text.SimpleDateFormat**

**SimpleDateFormat sd = new SimpleDateFormat**

**("yyyy – MM – dd E HH:mm:ss");  
sd.format(new Date());**

**Calendar cal = Calendar.getInstance();**

**sd.format (cal.getTime())**

**Date-> import 자동 추가 메뉴**

**12장 멀티스레드**

**프로그램 – 컴퓨터 작업 순서 정의 절차**

**문법 언어마다 다르다**

**프로세스 – cpu 가 현재 실행중인 프로그램**

**줌/pdf뷰어/탐색기/크롬**

**스레드(thread) – 프로세스를 구성하는 작업 단위**

**예)**

**크롬 프로세스**

**1> 음악 사이트 플레이 스레드**

**2> 뉴스 게시판 스레드**

**3> 다운로드 사이트 스레드**

**cpu 코어 1개 – 스레드**

**- windows – cpu**

**cpu = 여러개 코어 구성**

**12.1 – 12.4**

**예)**

**크롬 프로세스**

**1> 음악 사이트 플레이 스레드**

**2> 뉴스 게시판 스레드**

**3> 다운로드 사이트 스레드**

**single thread - 1번에 1개 스레드 실행 -->**

**플레이 종료 - 뉴스 종료 – 다운로드**

**멀티스레드 – multithread**

**1번에 여러개 스레드 동시 실행**

**( 모든 컴퓨터 1개 cpu /여러개 cpu)**

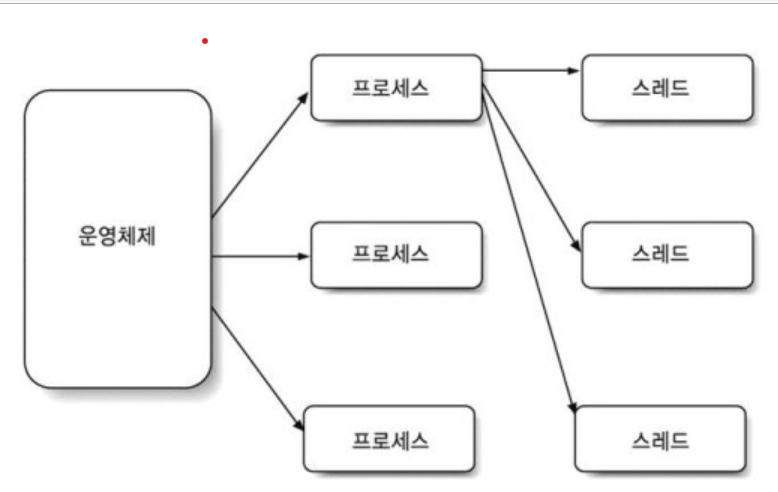
**플레이 시작 – 일시중단 –**

**뉴스 시작- 일시중단 –**

**다운로드 시작 – 일시중단**

**- 플레이 재시작 - ....**

**==> 번갈아가며 작업 = 빠르게 실행 = 동시 실행**

****

**- 자바 멀티스레드 구현 방법**

**java.lang.Thread 클래스 이용**

**1> 멀티스레드 클래스 정의**

**class A extends Thread {**

**public void run(){오버라이딩 }**

**+ ..........**

**}**

**2> 멀티스레드 객체 생성**

**A a1 = new A(); or**

**Thread a1 = new A();**

**3> 멀티스레드 메소드 호출 – 실행**

**a1.start();// run 멀티스레드**

|  |  |
| --- | --- |
| **run()** | **오버라이딩!!!!** |
| **start()** | **호출- run 시작** |
| **setName()** | **멀티스레드이름 설정** |
| **getName()** | **멀티스레드이름 조회** |
| **setPriority**  **(Thread.MAX\_PRIORITY)**  **setPriority(10)** | **1- 5(기본)- 10**  **Thread.MAX\_PRIORITY**  **Thread.MIN\_PRIORITY**  **Thread.NORM\_PRIORITY** |
| **10/5/1<-getPriority()** | **우선순위 조회** |

**WINDOWS + CPU CORE -->**

**또는**

**java.lang.Runnable 인터페이스 이용**

**1> 멀티스레드 클래스 정의**

**class B extends C implements Runnable {**

**//Runnable 상속 오버라이딩 의무화**

**public void run(){오버라이딩 }**

**+ ..........**

**}**

**}**

**2> 멀티스레드 객체 생성**

**Runnable b1 = new B();**

**Thread tb = new Thread(b1);**

**3> 멀티스레드 메소드 실행**

**tb.start();**

**종료**

**생성-**

**new Thread()**

**new Thread(Runnable)**

**run완료**

**start()**

**대기-**

**run() 수행 대기열**

**[ ,main, t2 ]**

**스레드 스케쥴링 순서대로**

**1. 우선순위 높은 것**

**2. 시간 할당 –자동결정**

**기본 : 5->10변경->1변경**

**(1-10 설정가능)**

**10 : t1(10초-중단) t2(10초-중단)**

**5 : t3(5초...)**

**1: t4(3초.. t5(3초**

**실행-**

**run() 수행**

**T2**

**run(){data 20 저장 5번 반복}**

**DATA**

**T1**

**run(){ data 10 저장 5번 반복}**

**DATA – 동기화 객체 생성**

**시작-....-종료**

**lock 해제**

**T1.RUN(LOCK 설정) :**

**[10, 10, 10, 10, 10]**

**T1..run종료하면서 LOCK 해제**

**lock**

**T1.start()**

**T2.start()**

**2개 이상 멀티스레드가 동시에 1개 객체 공유하면서 작업하는 환경에서**

**공유 객체 보호 = 1번에 1개 스레드만 접ㄱ슨 = LOCK**

**=synchronized**