

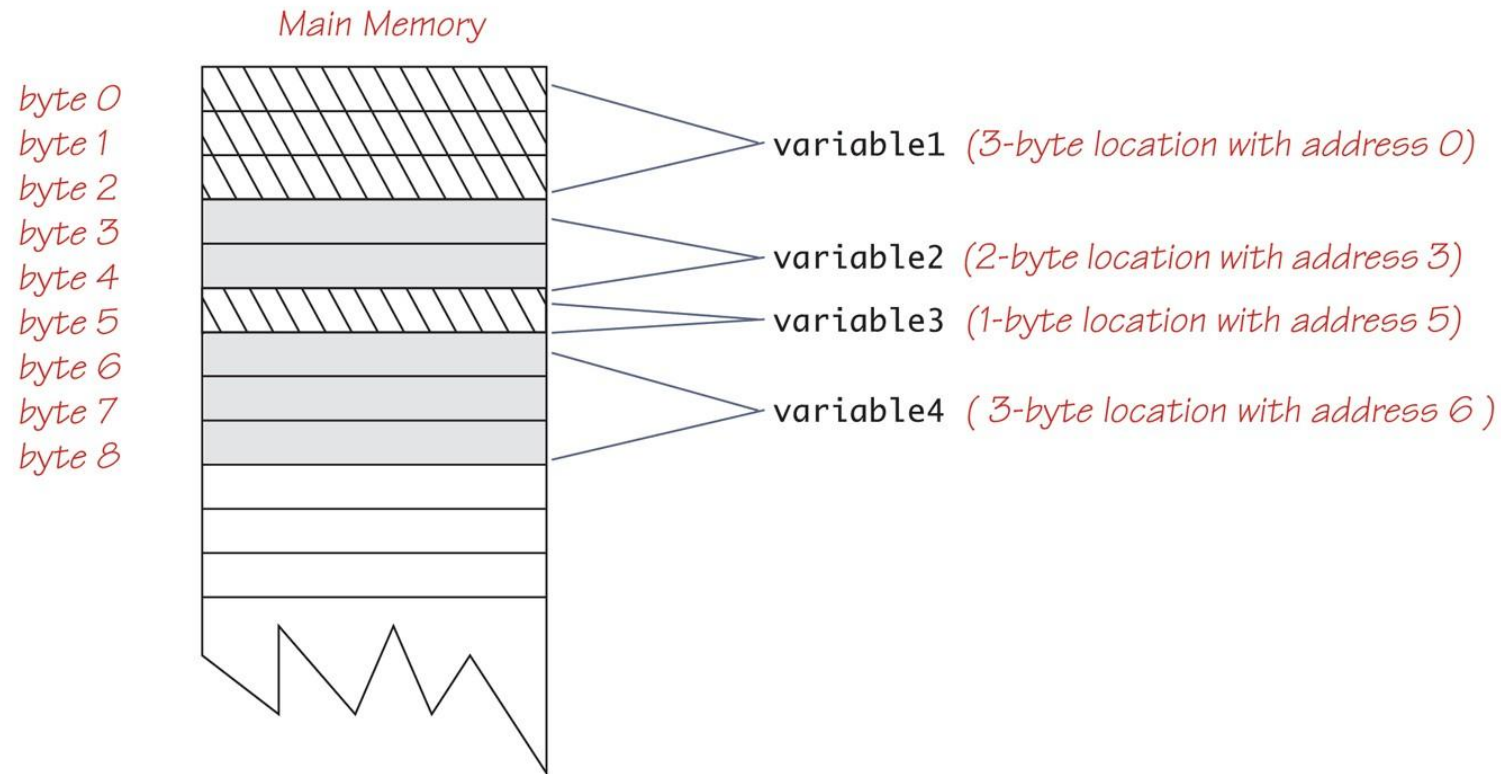
Object – Oriented Programming

LAB #5. CLASS 3

Variables and Memory

- 메인 메모리는 데이터를 byte 단위로 연속적으로 저장하는데 사용된다.
- 첫번째 byte의 주소는 데이터의 주소이다.

Display 5.10 Variables in Memory



Variables and Memory

- 기본형 타입은 실제 값을 메모리에 저장한다.

```
int i = 20;
```

```
char c = 'h';
```

```
short sh = 10;
```

0	20
1	
2	
3	
4	h
5	
6	10
7	
8	
9	
10	

Variables and Memory

```
int a, b;  
a = 20;  
b = a;  
System.out.println("a: " + a);  
System.out.println("b: " + b);  
b = 30;  
System.out.println("a: " + a);  
System.out.println("b: " + b);
```

a	20
b	20



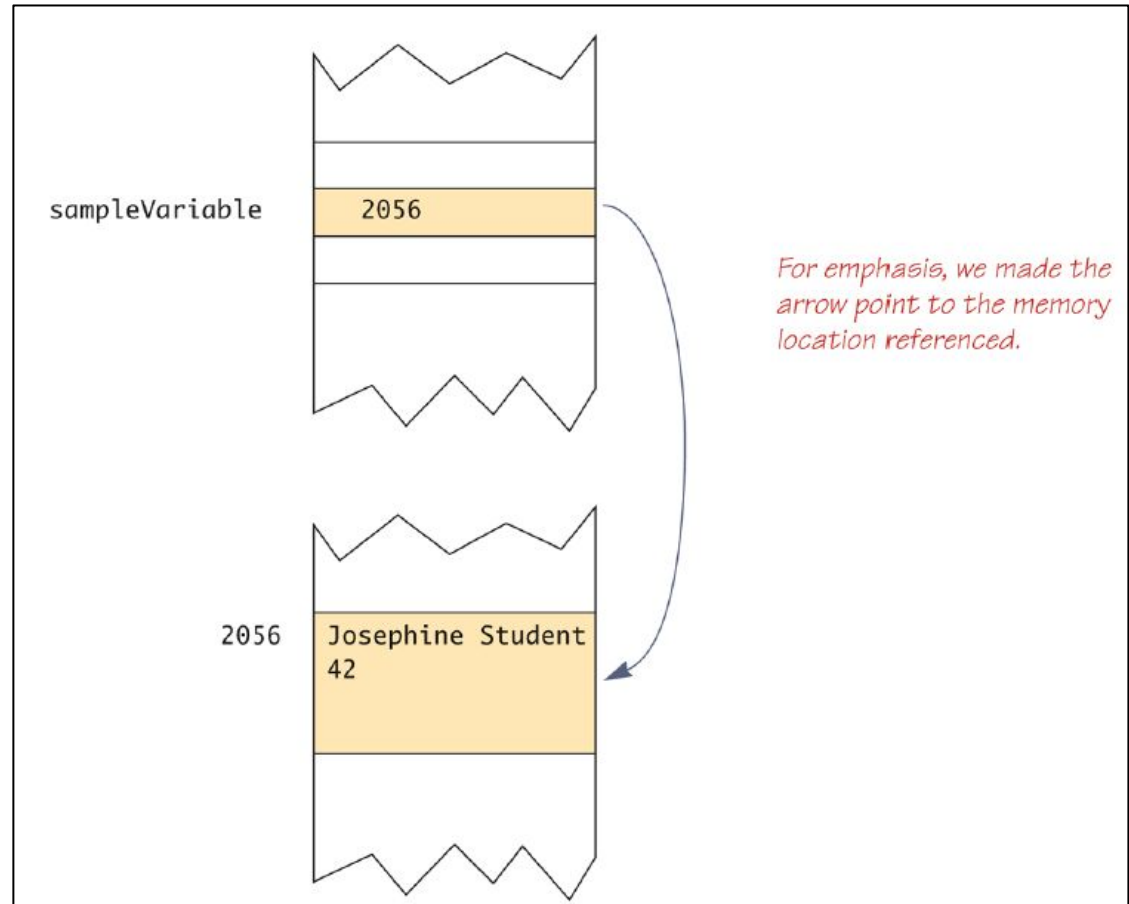
a	20
b	30

Variables and Memory

- 클래스 타입은 memory address를 메모리에 저장한다.

<https://blog.hexabrain.net/104>

```
ToyClass sampleVariable;  
sampleVariable =  
    new ToyClass("Josephine Student", 42);
```



ToyClass

```
public class ToyClass {
    private String name;
    private int number;

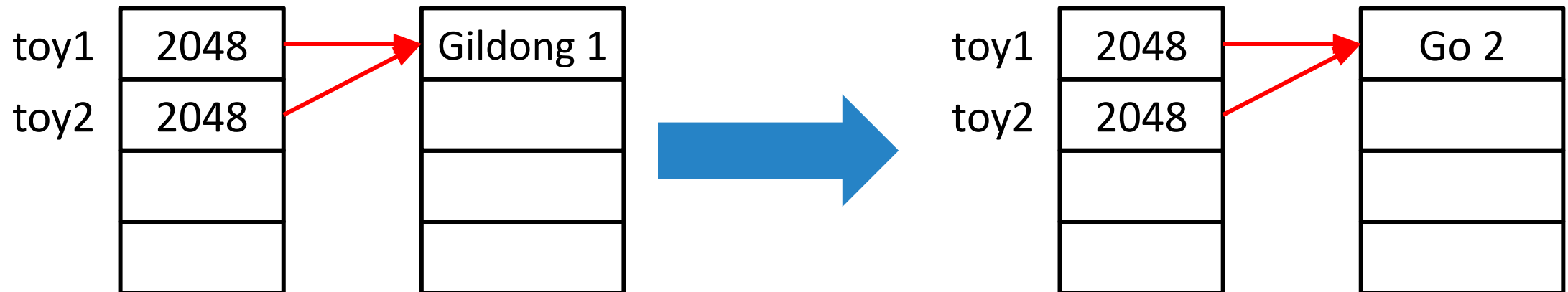
    public ToyClass(String initialName, int initialNumber){
        name = initialName;        number = initialNumber;
    }
    public ToyClass(){
        name = "No name yet.";    number = 0;
    }
    public void set(String newName, int newNumber){
        name = newName;          number = newNumber;
    }
    public String toString(){
        return (name + " " + number);
    }

    public static void changer(ToyClass aParameter){
        aParameter.name = "Hot Shot";    aParameter.number = 42;
    }

    public boolean equals(ToyClass otherObject){
        return ((name.equals(otherObject.name)) && (number == otherObject.number) );
    }
}
```

Variables and Memory

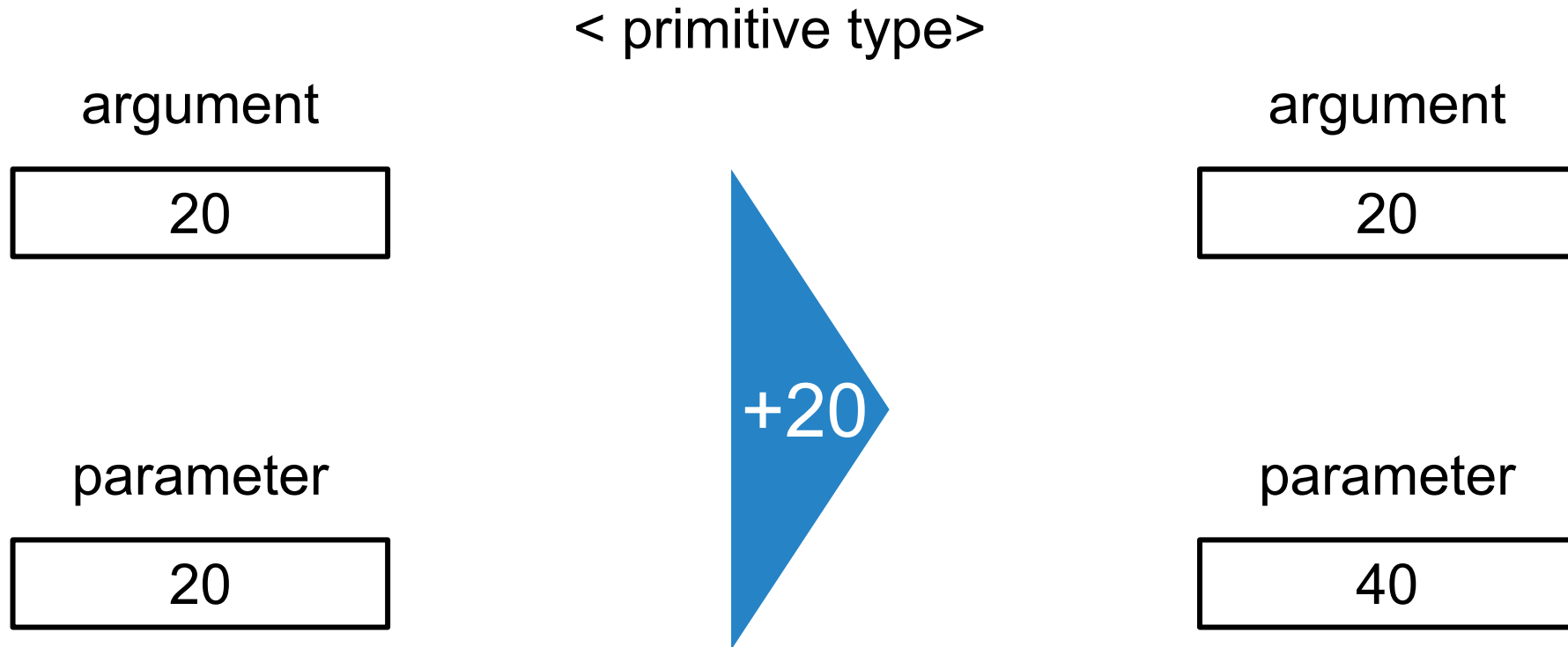
```
ToyClass toy1, toy2;  
toy1 = new ToyClass("Gildong", 1);  
toy2 = toy1;  
System.out.println("toy1: " + toy1);  
System.out.println("toy2: " + toy2);  
toy2.set("Go", 2);  
System.out.println("toy1: " + toy1);  
System.out.println("toy2: " + toy2);
```



Parameters

- Java의 parameter는 모두 call-by-value parameter
- 때문에, primitive type의 parameter의 값 변화는 argument의 값에 영향을 주지 않음
- 반면에, class type의 parameter의 값 변화는 argument의 값에 영향을 미침

Parameters



primitive type의 parameter의 값 변화는 argument의 값에 영향을 주지 않음

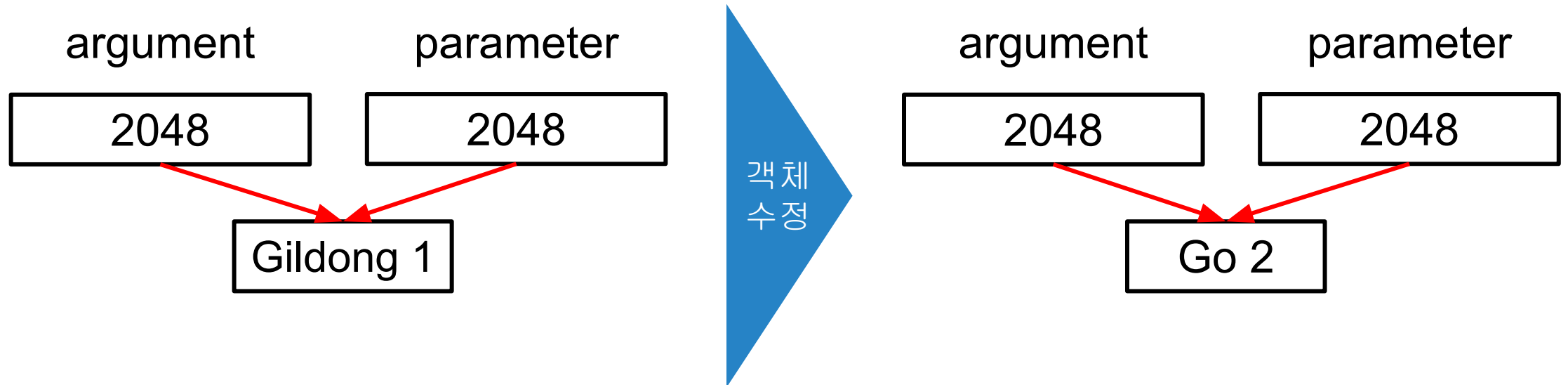
Parameters

```
public static void add20(int aParameter) {  
    System.out.println("aParameter: " + aParameter);  
    aParameter += 20;  
    System.out.println("aParameter: " + aParameter);  
}
```

```
int a = 20;  
System.out.println("a: " + a);  
add20(a);  
System.out.println("a: " + a);
```

Parameters

< class type >



class type의 parameter의 값 변화는 argument의 값에 영향을 미침

Parameters

```
public static void changer(ToyClass aParameter) {  
    aParameter.name = "Hot Shot";  
    aParameter.number = 42;  
}
```

```
ToyClass toy1 = new ToyClass("Gildong", 1);  
ToyClass.changer(toy1);  
System.out.println("toy1: " + toy1);
```

'==' and 'equals'

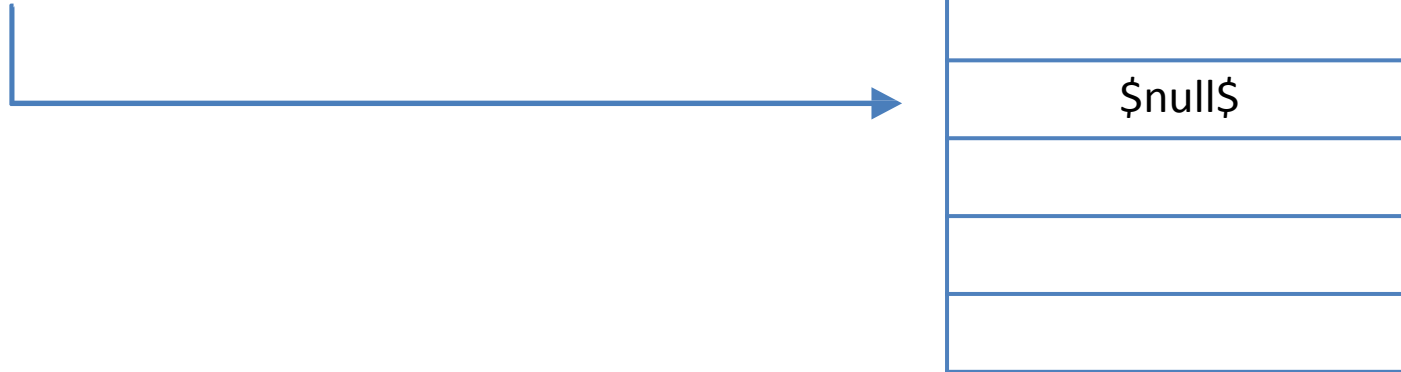
- Class type에서 '=='는 단순히 객체의 address를 비교
- 객체를 비교하기 위해서는 equals 사용

```
ToyClass toy1, toy2, toy3;  
toy1 = new ToyClass("Gildong", 1);  
toy2 = toy1;  
toy3 = new ToyClass("Gildong", 1);  
  
System.out.println(toy1 == toy2);  
System.out.println(toy2 == toy3);  
System.out.println(toy1.equals(toy3));
```

null

- 클래스 유형의 변수에 할당될 수 있는 특수한 상수

- `String str = null;`



- Null은 객체가 아니다.
- Null은 메모리 주소와 유사하며 변수에 null이 포함되어 있는지 테스트 하려면 `==`, `!=` 연산자를 사용해야 한다.
 - `if (str == null)`

new

- 객체를 초기화하는 생성자를 호출하고 생성된 객체의 메모리의 주소를 반환한다.
 - o 이 메모리의 주소를 클래스 유형의 변수에 할당한다.
- Anonymous Object : new 객체로 생성하여 반환 받은 주소를 변수에 할당하지 않는 것

```
if (variable1.equals(new ToyClass("Joe", 42)))  
    System.out.println("Equal");  
else  
    System.out.println("Not equal");
```

Date

```
public class Date {
    private String month;
    private int day;
    private int year;

    public Date(String month, int day, int year) {
        setDate(month, day, year);
    }

    public void setDate(String month, int day, int year) {
        this.month = month;
        this.day = day;
        this.year = year;
    }

    public String toString() {
        return (month + " " + day + ", " + year);
    }

    public boolean equals(Date otherDate) {
        return ((month.equals(otherDate.month)) && (day == otherDate.day) && (year == otherDate.year));
    }
}
```


Copy Constructors

- 기존의 생성한 객체를 복사하기 위한 생성자
- 클래스와 동일한 타입의 매개변수를 받는 생성자

```
public Date(Date aDate) {  
    if(aDate == null) { //Not a real date.  
        System.out.println("Fatal Error.");  
        System.exit(0);  
    }  
  
    this.month = aDate.month;  
    this.day = aDate.day;  
    this.year = aDate.year;  
}
```

Copy Constructors

```
Date date1 = new Date("April", 1, 2021);  
Date date2 = date1;  
Date date3 = new Date(date1);  
  
System.out.println(date1);  
System.out.println(date2);  
System.out.println(date3);  
  
date1.setDate("January", 1, 2022);  
  
System.out.println(date1);  
System.out.println(date2);  
System.out.println(date3);
```

Person

```
public class Person {  
    private String name;  
    private Date born;  
    private Date died;  
  
    public Person(String name, Date born, Date died) {  
        this.name = name;  
        this.born = born;  
        this.died = died;  
    }  
  
    public String toString() {  
        String diedString;  
        if (died == null)  
            diedString = ""; // Empty string  
        else  
            diedString = died.toString();  
        return (name + ", " + born + "-" + diedString);  
    }  
}
```

Person

```
public boolean equals(Person otherPerson) {  
    if (otherPerson == null)  
        return false;  
    else if (died == null)  
        return (name.equals(otherPerson.name)  
                && born.equals(otherPerson.born)  
                && otherPerson.died == null);  
    else  
        return (name.equals(otherPerson.name)  
                && born.equals(otherPerson.born)  
                && died.equals(otherPerson.died));  
}  
  
public Date getBorn() {  
    return born;  
}  
}
```

Person – Copy Constructors(Dangerous)

```
public Person(Person aPerson){  
    if (aPerson == null) { //Not a real date.  
        System.out.println("Fatal Error.");  
        System.exit(0);  
    }  
  
    name = aPerson.name;  
    born = aPerson.born;  
    if(died == null)  
        died = null;  
    else  
        died = aPerson.died;  
}
```

Person – Copy Constructors(Dangerous)

- Copy한 객체의 변화가 다른 객체에도 영향을 미침

```
Person person1, person2;  
person1 = new Person("Go", new Date("January", 1, 2021), null);  
person2 = new Person(person1);  
System.out.println(person1);  
System.out.println(person2);  
person1.getBorn().setDate("April", 1, 2021);  
System.out.println(person1);  
System.out.println(person2);
```

Person – Copy Constructors(Safe)

```
public Person(Person aPerson){  
    if (aPerson == null) { //Not a real date.  
        System.out.println("Fatal Error.");  
        System.exit(0);  
    }  
  
    name = aPerson.name;  
    born = new Date(aPerson.born);  
    if(died == null)  
        died = null;  
    else  
        died = new Date(aPerson.died);  
}
```

Person – getter(Dangerous)

– private 변수임에도 불구하고 수정 가능

```
Person person3;  
person3 = new Person("Go", new Date("January", 1, 2021), null);  
System.out.println(person3);  
person3.getBorn().setDate("April", 1, 2021);  
System.out.println(person3);
```


Person – getter(Safe)

```
public Date getBorn() {  
    return new Date(born);  
}
```

Deep Copy vs Shallow Copy

- Deep Copy : 원본과 공통되는 참조가 없는 복사

```
public Date getBorn() {  
    return new Date(born);  
}
```

- Shallow Copy : 그 외의 복사
이 경우의 두 변수가 같은 참조를 가질 수가 있어, 보안상의 위험이 있다.

```
public Date getBorn() {  
    return born;  
}
```

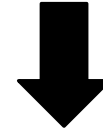
*LocalDate의 경우 예외(위 예시는 Date사용시)

<https://stackoverflow.com/questions/51842569/best-way-to-clone-java-time-localdatetime>

Package

- 서로 관련 있는 클래스들의 모음
- 프로젝트를 편리하게 관리하도록 함
- 다른 라이브러리들끼리 구분 가능
- import 키워드를 사용하여 클래스를 간단히 사용 가능

```
public class Program {  
    public static void main(String[] args) {  
        java.util.Scanner scan = new java.util.Scanner(System.in);  
    }  
}
```



```
import java.util.Scanner;  
  
public class Program {  
    public static void main(String[] args) {  
        Scanner scan = new Scanner(System.in);  
    }  
}
```

Package – Name Clash

```
package kr.ac.hanyang1;  
  
public class SayHello {  
    public void sayPackage() {  
        System.out.println("This is hanyang1.");  
    }  
}
```

```
package kr.ac.hanyang2;  
  
public class SayHello {  
    public void sayPackage() {  
        System.out.println("This is hanyang2.");  
    }  
}
```

```
packageTest  
└─ JRE System Library [JavaSE-13]  
    └─ src  
        └─ kr.ac.hanyang1  
            └─ SayHello.java  
        └─ kr.ac.hanyang2  
            └─ SayHello.java  
        └─ kr.ac.hanyang3  
            └─ Tester.java  
        └─ module-info.java
```

Package – Name Clash

```
package kr.ac.hanyang3;

import kr.ac.hanyang1.*;
import kr.ac.hanyang2.*;

public class Tester {

    public static void main(String[] args) {

        kr.ac.hanyang1.SayHello test1 = new kr.ac.hanyang1.SayHello();
        kr.ac.hanyang2.SayHello test2 = new kr.ac.hanyang2.SayHello();

        test1.sayPackage();
        test2.sayPackage();

    }
}
```

Date

- java.util 패키지에 있는 클래스
- 날짜와 시간에 관한 정보를 표현하는 클래스
- 현재 JAVA 버전에서는 사용되지 않음
→ Calendar 클래스로 대체

```
Date today = new Date();
System.out.println(today);

int year = today.getYear()+1900;
System.out.println(year);

int month = today.getMonth()+1;
System.out.println(month);

int day = today.getDay()+10;
System.out.println(day);

int hours = today.getHours();
System.out.println(hours);

int minutes = today.getMinutes();
System.out.println(minutes);
```

Calendar

- java.util 패키지에 있는 클래스
- 날짜와 시간에 관한 정보를 표현하는 클래스
- Date 클래스의 대체물로 나온 클래스
- Month 는 1을 더해줘야 정상적인 값이 나옴
- Immutable하지 않음

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

int year = cal.get(Calendar.YEAR);
System.out.println(year);

int month = cal.get(Calendar.MONTH) + 1;
System.out.println(month);

int day = cal.get(Calendar.DAY_OF_MONTH);
System.out.println(day);

int hours = cal.get(Calendar.HOUR_OF_DAY);
System.out.println(hours);

int minutes = cal.get(Calendar.MINUTE);
System.out.println(minutes);
```

LocalDateTime

- jdk 1.8부터 지원
- java.time 패키지에 있는 클래스
- 날짜와 시간에 관한 정보를
표현하는 클래스

<https://jeong-pro.tistory.com/163>

```
LocalDateTime currentTime =  
    LocalDateTime.now();  
LocalDateTime myDateTime  
    = LocalDateTime.of(2021, 4, 1, 15, 00);  
  
int year = myDateTime.getYear();  
System.out.println(year);  
  
Month month1 = myDateTime.getMonth();  
System.out.println(month1);  
int month2 = myDateTime.getMonthValue();  
System.out.println(month2);  
  
int day1 = myDateTime.getDayOfMonth();  
System.out.println(day1);  
DayOfWeek day2 = myDateTime.getDayOfWeek();  
System.out.println(day2);  
int day3 = myDateTime.getDayOfYear();
```


LocalDateTime

```
LocalDateTime currentTime = LocalDateTime.now();
LocalDateTime myDateTime = LocalDateTime.of(2021, 4, 1, 15, 00);

System.out.println(currentTime.isAfter(myDateTime));
System.out.println(currentTime.isBefore(myDateTime));
System.out.println(currentTime.isEqual(myDateTime));

LocalDateTime minusDateTime = myDateTime.minusDays(5);
System.out.println(myDateTime);
System.out.println(minusDateTime);
// minusDays, Weeks, Months, Years ...
// plusDays, Weeks, Months, Years ...
```

LocalDate

- jdk 1.8부터 지원
- java.time 패키지에 있는 클래스
- 날짜에 관한 정보를 표현하는 클래스
- 이 외에도
plusMonths, isBefore
등의 메소드가 있음.

```
LocalDate currentDate = LocalDate.now();
LocalDate myDate = LocalDate.of(2021, 4, 1);

int year = myDate.getYear();
System.out.println(year);

Month month1 = myDate.getMonth();
System.out.println(month1);
int month2 = myDate.getMonthValue();
System.out.println(month2);

int day1 = myDate.getDayOfMonth();
System.out.println(day1);
DayOfWeek day2 = myDate.getDayOfWeek();
System.out.println(day2);
int day3 = myDate.getDayOfYear();
System.out.println(day3);
```

LocalTime

- jdk 1.8부터 지원
- java.time 패키지에 있는 클래스
- 시간에 관한 정보를
표현하는 클래스

```
LocalTime currentTime = LocalTime.now();
LocalTime myTime = LocalTime.of(15, 0);
System.out.println(myTime);
LocalTime myTime2 = LocalTime.of(15, 0, 15, 28);
System.out.println(myTime2);

int hour = myTime.getHour();
System.out.println(hour);
int minute = myTime.getMinute();
System.out.println(minute);
int second = myTime2.getSecond();
System.out.println(second);
int nano = myTime2.getNano();
System.out.println(nano);
```

실습 과제

1. `account` 패키지를 추가하고 `Account` 클래스 생성
2. 다음 내용을 참고하여 `Account` 클래스를 구현

Account	
Private	<ul style="list-style-type: none">- name : String- yearlyInterest : double- balance: double- created: LocalDate
Public	<ul style="list-style-type: none">+ Account(name : String, yearlyInterest : double, created: LocalDate)+ getBalance() : double+ getCreated() : LocalDate+ increaseYearlyInterest(byPercent : int) : void+ receiveIncome (double income) : void+ receiveInterest () : void+ toString() : String

실습 과제

3. Account 클래스의 Method 설명

- Account 생성자에서는
이름(name)과 연이자(yearlyInterest), 가입일(created)을 매개변수로 전달받고 balance는 0으로 초기화한다.
- 잔고(balance)를 반환하는 getter 생성. (getBalance)
- 가입일을 반환하는 getter 생성.
- 연이자를 증가시켜주는 Method를 생성한다. (increaseYearlyInterest)
 - 매개변수로 받은 값을 퍼센트 포인트로 생각하고, 현재 yearlyInterest 값을 해당 값만큼 증가시킨다.
 - ex) 매개변수의 값이 1이고 연이자 4(%)일 경우, 1을 더하여 5로 만든다.
- 계좌에 수익을 받는 Method를 생성한다. (receiveIncome)
 - balance 에 파라미터로 받는 income 을 더한다.
- 복리 이자를 받는 Method를 생성한다. (receiveInterest)
 - 연이자율을 기준으로 받는 이자를 12로 나눠서 더한다. (잔고*이자율 / 12)
- toString Method를 생성한다.
 - return 값 예시 - 이름 : Lee 연이자 : 5 잔고 : 0.0 가입일: 2022-01-01

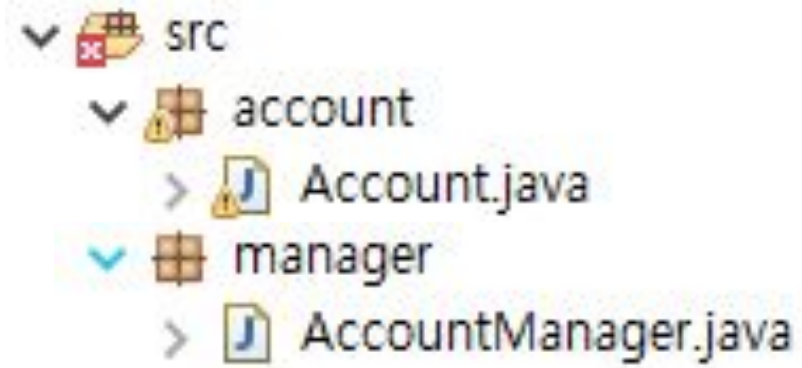
실습 과제

4. AccountManager Class 생성 (**manager** 패키지에 위치) 이 클래스에 Main Method를 설정한다.

- 가입일(created)을 LocalDate.of()를 사용해서 생성한다. (가입일 : 2021-12-01)
- 사용자 계좌를 생성한다. (이름 : 자신의 성 / 연이자 : 5 / 가입일: created)
- 해당 계좌의 모든 정보를 출력한다. (ex_ 이름 : Kim, 연이자: 5, 잔고: 0, 가입일: 2021-12-01)
- 잔고가 10000(1만)이 될 때까지 아래 활동을 반복하는 반복문을 생성한다.
 - 매달 지날수록 사용자는 이자와 소득을 받는다. (Account의 receiveIncome(100), receiveInterest() Method 사용)
- 1년 이상 가입한 고객에 한해 매 1월 이벤트를 진행한다.
 - 당첨율 10퍼센트. 100만원 상당의 현금성 상품권 지급, 계좌로 입금.(receiveIncome(100))
 - 이벤트 당첨자의 경우, 당첨 정보를 출력한다. (ex_ 이벤트 당첨!)
 - 당첨율 구현 예시: 0~9까지의 난수를 생성한 후 0~9 중 하나의 숫자를 만족하면 실행
- 가입한지 3년이 지났을 경우 이자율을 한번만 증가시킨다.(각자 원하는 숫자)
 - Account의 increaseYearlyInterest(bypercent : int) Method 사용
 - 메시지를 출력한다. (ex_ 가입 후 3년이 지나서 이자율이 2% 인상되었습니다.)
 - 한번만 증가시키는 방법: 힌트_ int flag 변수를 추가로 사용한다.
- 잔고가 10000(1만)이상 일 때, 해당 계좌의 모든 정보를 출력한다.
(ex_ 이름: Kim, 연이자: 7, 잔고: 10000, 가입일: 2021-12-01)

실습 과제

예시 구조



실습 과제

예시 출력

이름: Kim, 연이자: 5.0, 잔고: 0.0, 가입일: 2021-12-01

가입 후 3년이 지나서 이자율이 2% 인상되었습니다.

이벤트 당첨!

이름: Kim, 연이자: 7.0, 잔고: 10018.158683210642, 가입일: 2021-12-01, 1억 모으기 끝: 2028-09-29

퀴즈 공지!