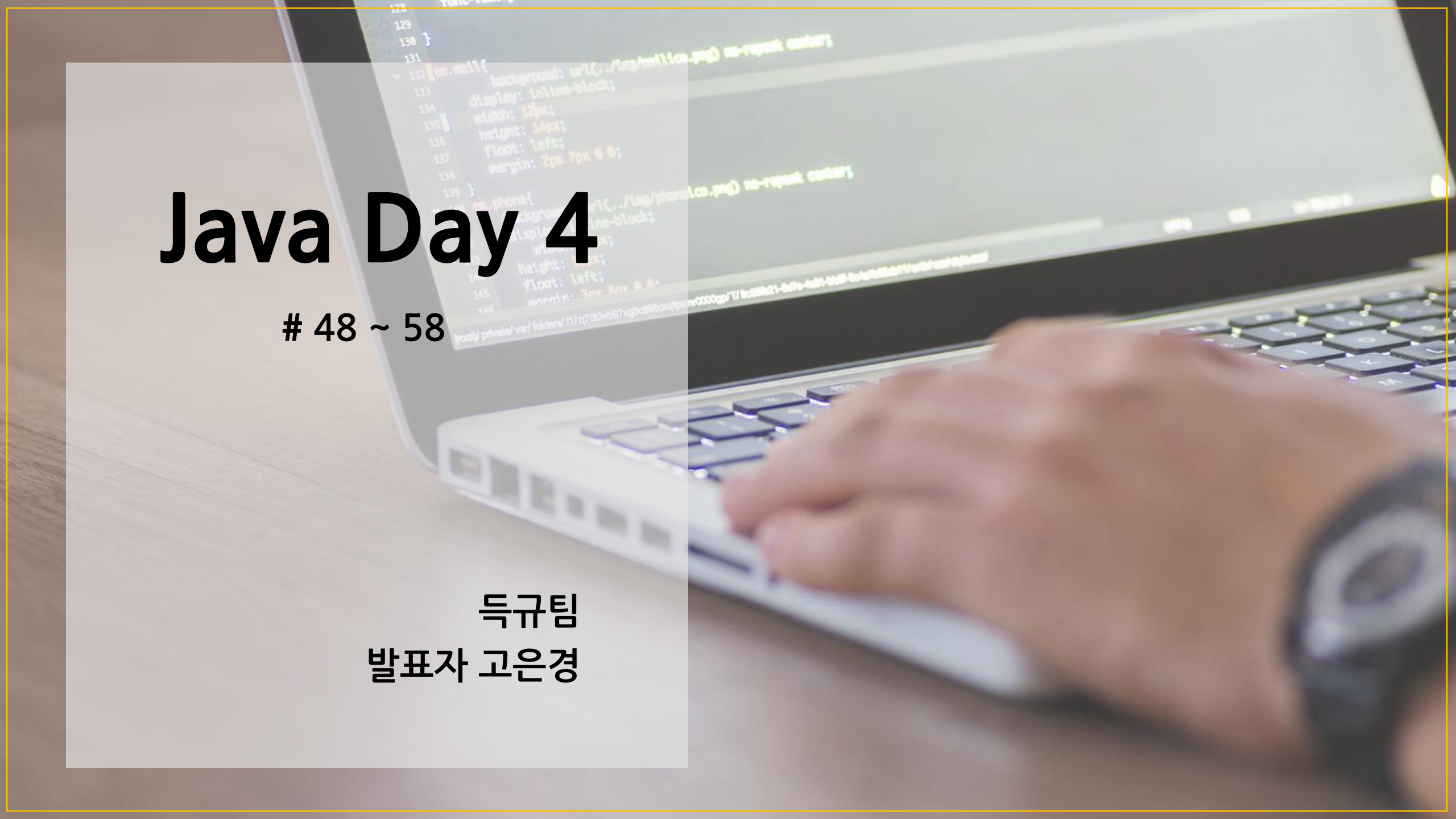


Java Day 4

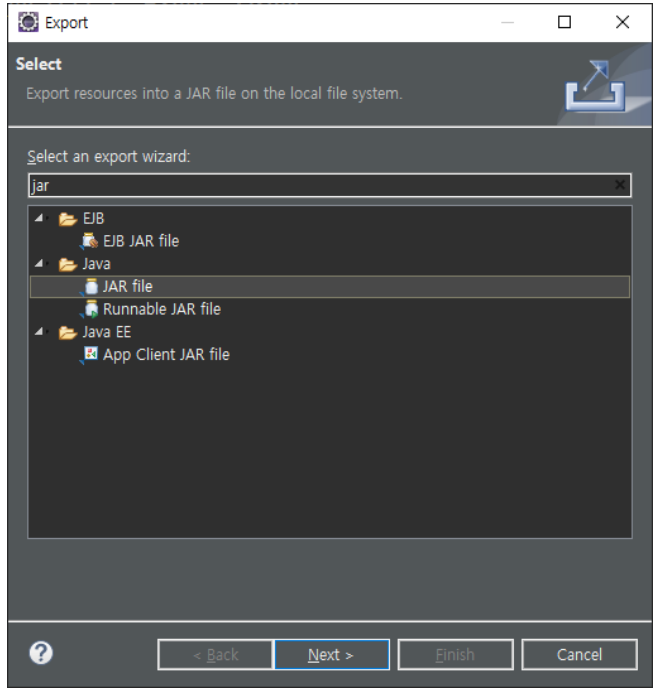
48 ~ 58

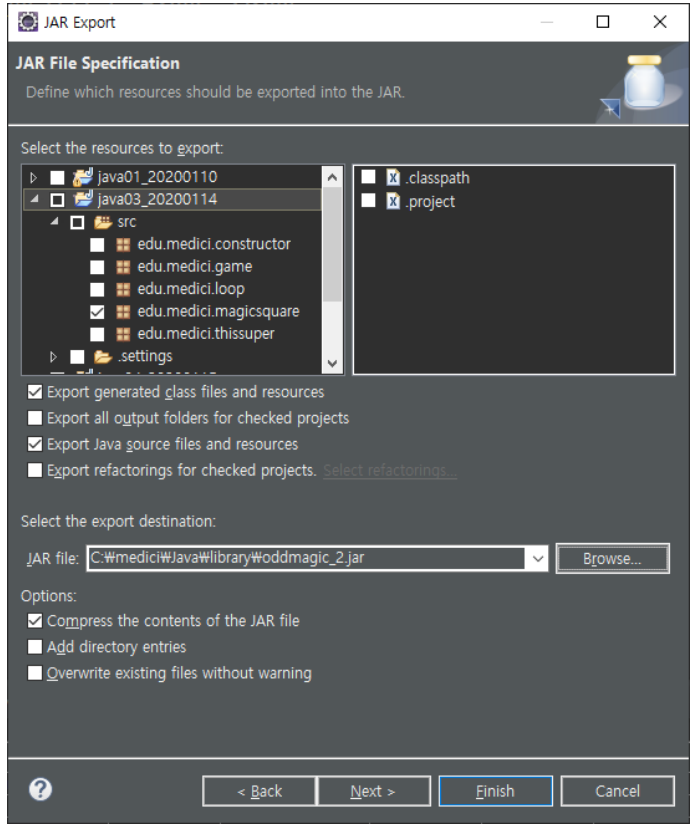
득규팀
발표자 고은경

48 jar (java archive)

- java로 만들어진 프로그램을 압축해서 라이브러리 형태로 사용할 수 있다.
- 다른 사람들에게 배포해서 사용하게 할 수 있다.
- jar파일을 만들어서 export(외부로 배포)하거나 jar파일을 만들어서 import(library로 사용)할 수 있다.

`export: export 원하는 프로젝트 이름에서 우클릭 -> export -> jar 타이핑 -> jar file`

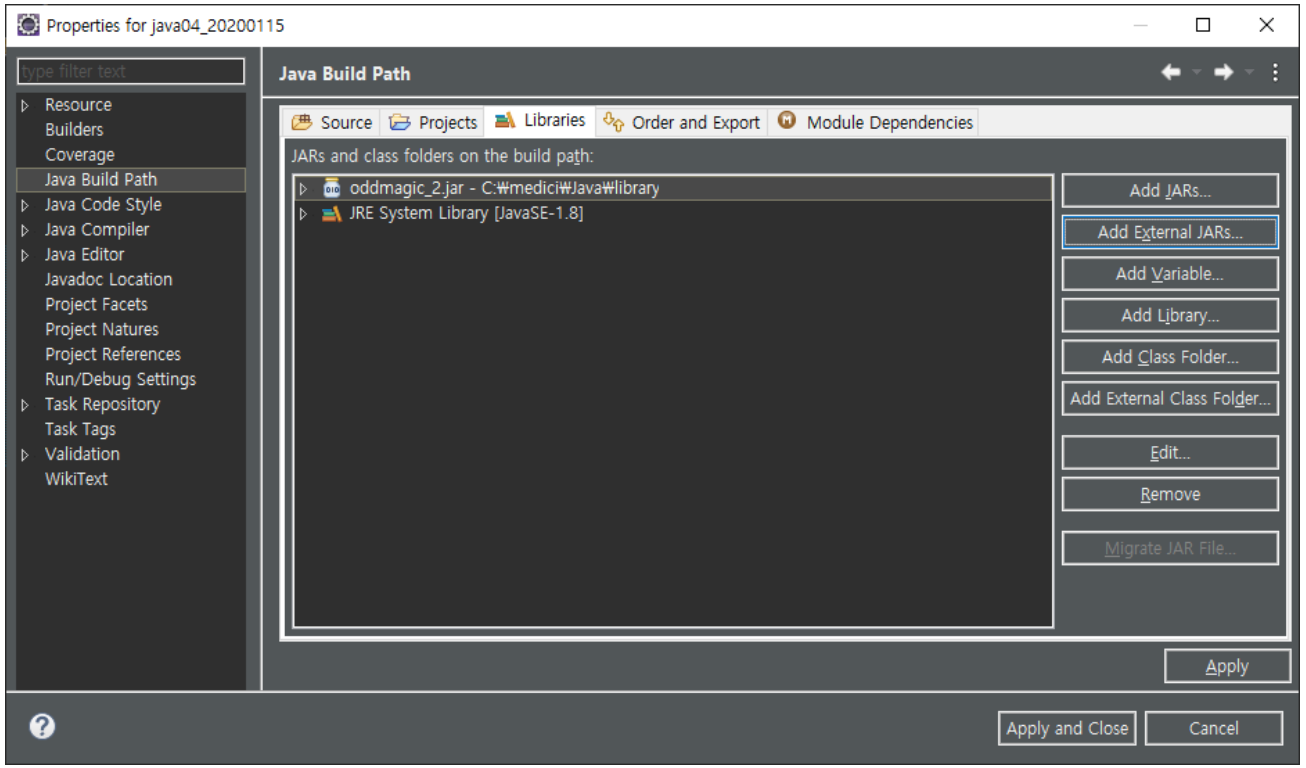


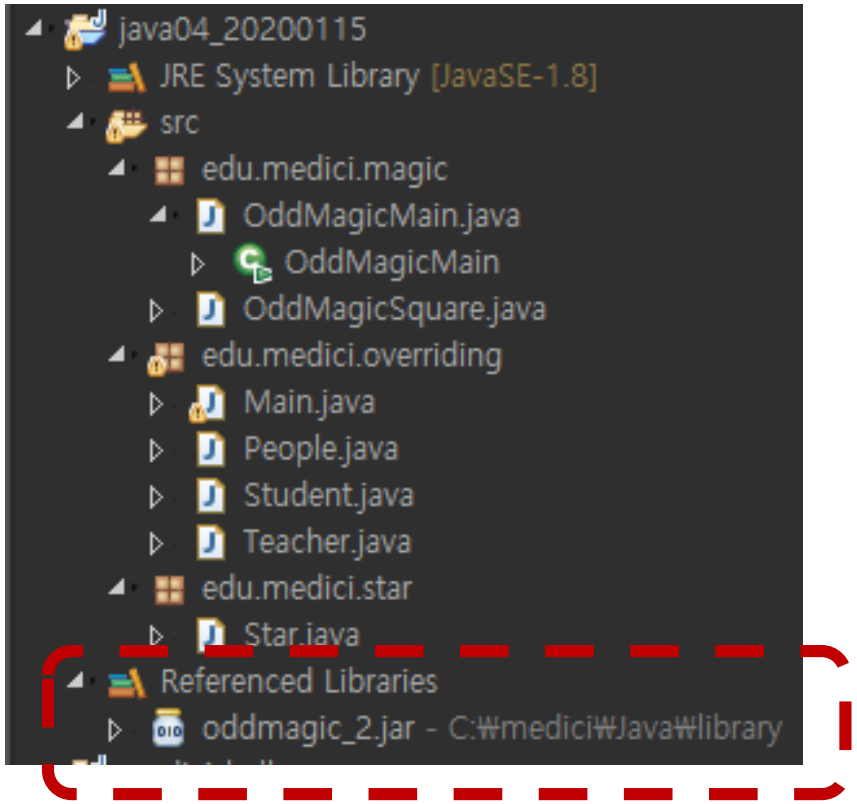


48 jar (java archive)

- java로 만들어진 프로그램을 압축해서 라이브러리 형태로 사용할 수 있다.
- 다른 사람들에게 배포해서 사용하게 할 수 있다.
- jar파일을 만들어서 export(외부로 배포)하거나 jar파일을 만들어서 import(library로 사용)할 수 있다.

`import: import 원하는 프로젝트 이름에서 우클릭 -> properties -> Java Build Path -> libraries -> Add External JARS`





49 import

- 다른 패키지에 있는 클래스 파일을 가져와서 사용할 때 적용하는 문법이다.
- package 다음에 올 수 있다.
- IDE에서 자동으로 추가를 해준다.
- 라이브러리를 추가하고 사용할 때도 import를 해줘야 라이브러리에 있는 파일을 사용할 수 있다.
- package명 + 클래스

```
import edu.medici.magic.OddMagicSquare
```

- 패키지에 있는 클래스를 다 가져오고 싶으면 `*`를 사용한다.
  - 메모리 문제가 발생할 수 있기 때문에 가능한 한 필요한 클래스만 import 하는 것이 좋다.

```
import edu.medici.magic.*
```

50 오버로딩 (메소드 오버로딩)

pdf p.60

- 변수의 타입, 개수가 다르면 다른 method로 인식함

```
public void eat();
public void eat(int bab);
public void eat(double bab);
```

51 형변환(casting)

pdf p.38

- 기본타입 형변환
  - 큰 <- 작 : auto promotion, up-casting, implicit(암시적)
  - 작 <- 큰 : down-casting (=casting), explicit(명시적)
    - (int)(Math.random()*45)
      - 소수점 잘리는 것 감안
- 참조타입 형변환
  - 부모 <- 자식 : auto promotion, up-casting, implicit(암시적)

```
// People - Student 상속 관계에서
People p = new People();
Student s = new Student();
p = s; // 부모 이름으로 자식을 받음
```

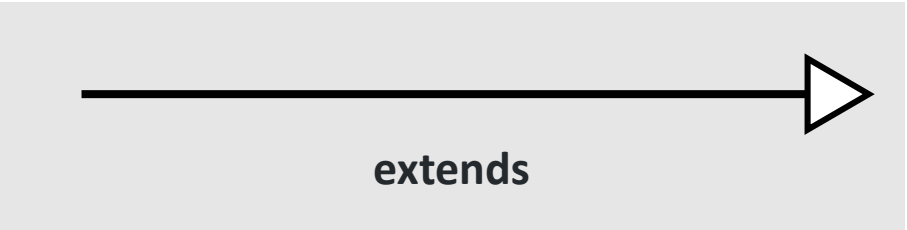
-
  - 자식 <- 부모 : (down-)casting, explicit(명시적)

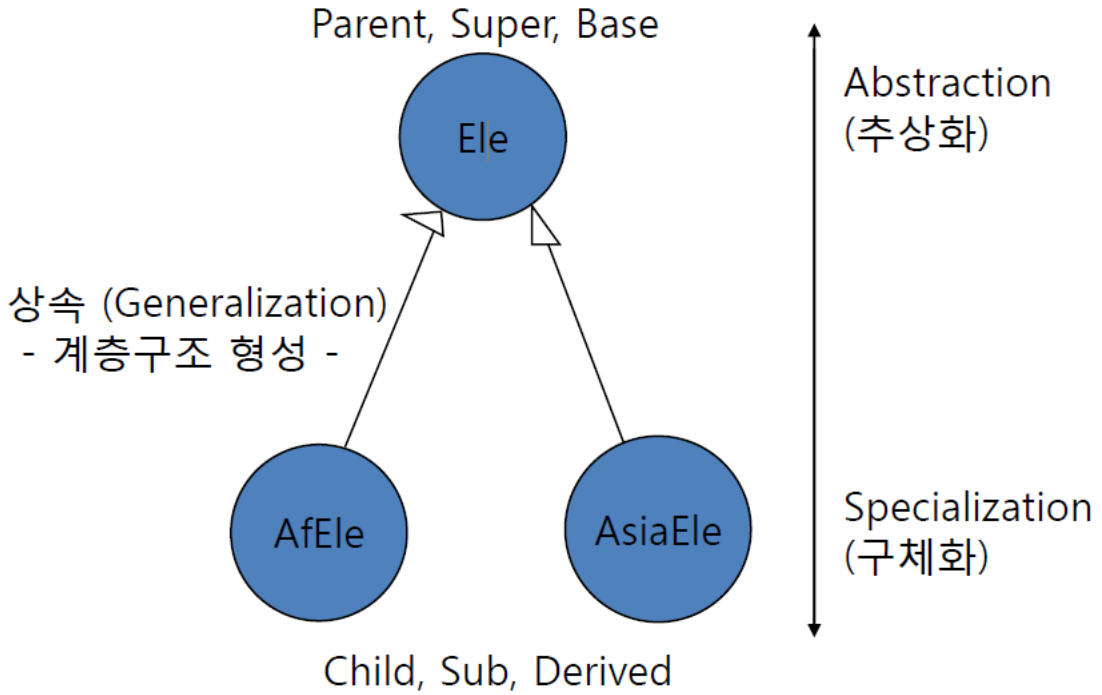
```
// People - Student 상속 관계에서
People p = new People();
Student s = new Student();
s = (S)p; // 자식 이름으로 부모를 받음
```

52 OOP 용어

pdf p.46

- 부모(parent, super), 자식(child, sub, derived)
- 계층구조(hierarchy)가 형성되어 있다.
  - 자식 -> 부모: 추상화(abstraction), 일반화(Generalization)
  - 부모 -> 자식: 구체화(Specialization)





53 boxing / unboxing

- boxing: 기본타입을 참조타입으로 바꾸는 것
- unboxing: 참조타입을 기본타입으로 바꾸는 것
- 일종의 캐스팅(타입 캐스팅)

```
int a = 10;
Integer aa = new Integer(a); // 기본 -> 참조
Integer aa2 = 10; // 기본 -> 참조
int c = aa2; // 참조 -> 기본
int c1 = aa.intValue(); // 참조 -> 기본

Object o = 10; // 기본 -> 참조   // Java의 최상위 클래스는 Object
int d = (Integer)o; // 참조 -> 기본
```

54 Wrapper Class

pdf p.70

- 참조타입을 기본타입으로 바꿀 때 사용하는 클래스
  - 기본 -> 참조할때도!
- 기본타입의 앞의 글자를 대문자로 만들면 클래스명이 된다.

```
Boolean,
Byte,
Character,
Short,
Integer,
Long,
Float,
Double
```

55 Integer.parseInt();

- 문자열을 int형으로 변환할 때 사용하는 Wrapper class 메소드

```
int passwd = Integer.parseInt("12345");
```

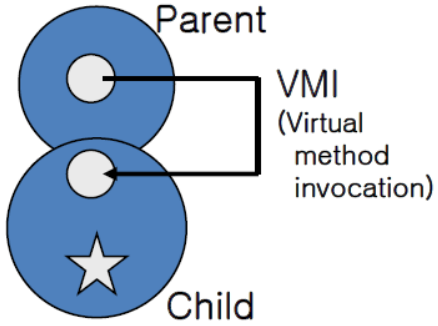
56 static / non-static

- static(메모리)
  - 객체를 생성하지 않고 바로 사용할 수 있다.
  - ex. `Math.PI`
- non-static
  - 객체를 생성해야 사용할 수 있다.
  - ex. `People p = new People();`
- non-static에서는 static을 사용할 수 있지만, static에서는 non-static을 사용할 수 없다.
- static에서 non-static을 사용하려면
  - 객체를 생성하거나
  - static 키워드를 붙이면 된다.
- member field, member method에 모두 사용이 가능하다.

57 overriding (오버라이딩)

pdf p.54

- 부모 자식 관계에서 부모에 있는 method와 자식에 있는 method가 동일하면 자식의 method가 호출되는 것



58 java.lang.Object의 4대 메소드

- boolean equals(Object o)
  - 객체끼리 같은지를 판단할 때 사용하는 메소드

```
A a = new A();
A a2 = new A();
System.out.println(a.equals(a2)); // false
// equals를 오버라이딩해서 값이 같으면 객체가 같다라고 판단하게 구현을 해주면 된다.
// String str = new String("사용자");
// String str2 = "사용자";
// str.equals(str2); //true
```

- String toString();
  - 문자열 타입으로 변경시켜주는 메소드
- int hashcode();
  - 객체의 해쉬코드를 가져오는 메소드
- Class getClass();
  - 객체의 package명 + 클래스이름을 가져오는 클래스 타입의 메소드

```
edu.medici.magic.OddMagicSquare
```

Code Review 1. Star

```java
package edu.medici.star;

public class Star {
    public void star1(int n) {
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < i+1; j++) {
                System.out.print("*");
            }
            System.out.println();
        }

    }
```

```java
public static void main(String[] args) {
    Star star = new Star();
     star.star1(4);
```

```
*
**
***
****
```

```java
public void star2(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < 2*i+1; j++) {
            System.out.print("*");
        }
        System.out.println();
    }
}
```

```java
public static void main(String[] args) {
    Star star = new Star();
     star.star1(4);
     star.star2(4);
    star.star3(8);
}
```

```
*
***
*****
*******
```

```java
public void star3(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n-1-i; j++) {
            System.out.print(" ");
        }
        for (int k = 0; k < 2*i+1; k++) {
            System.out.print("*");
        }
        System.out.println();
    }
}
```

```java
    public static void main(String[] args) {
        Star star = new Star();
//       star.star1(4);
//       star.star2(4);
        star.star3(4);
    }
}
```

```
   *
  ***
 *****
*******
```

Code Review 2. 마방진 검증

```java
// 마방진 로직이 제대로 수행되었는지 확인하는 메서드
public boolean isCheck() {
    boolean isT = true;
    int count = top+1; // n
    int[] mcheck = new int[2*count+2];

    for (int i = 0; i < count; i++) {
        for (int j = 0; j < count; j++) {
            mcheck[i] += magic[i][j];
            mcheck[count+i] += magic[j][i];
            if (i==j) {
                mcheck[2*count] += magic[i][j];
            }
            if (i+j == count-1) {
                mcheck[2*count+1] += magic[i][j];
            }
        }
    }

        // 값이 실제로 틀린 값이 있는지 체크
        for (int i = 1; i < mcheck.length; i++) {
            if (mcheck[0] != mcheck[i]) {
                isT = false;
                break;
            }
        }
        return isT;
    }

}
```

```java
package edu.medici.magic;

public class OddMagicMain {
    public static void main(String[] args) {
        OddMagicSquare odd = new OddMagicSquare(7);
        odd.make();
        odd.print();
        System.out.println(odd.isCheck());
    }


}
```

```
6       1       8
7       5       3
2       9       4
true
```

Code Review 3. 오버로딩/오버라이딩

- source -> Generate Constructor using Fields
    - 하고나서 습관적으로 default 생성자 만들기

```java
package edu.medici.overriding;

public class People {
    private String name;
    private int age;

    @Override
    public String toString() {
        return "People [name=" + name + ", age=" + age + "]";
    }

    public People() {}
    public People(String name, int age) {
        super();
        this.name = name;
        this.age = age;
    }
    // name
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    // age
    public int getAge() {
        return age;
```

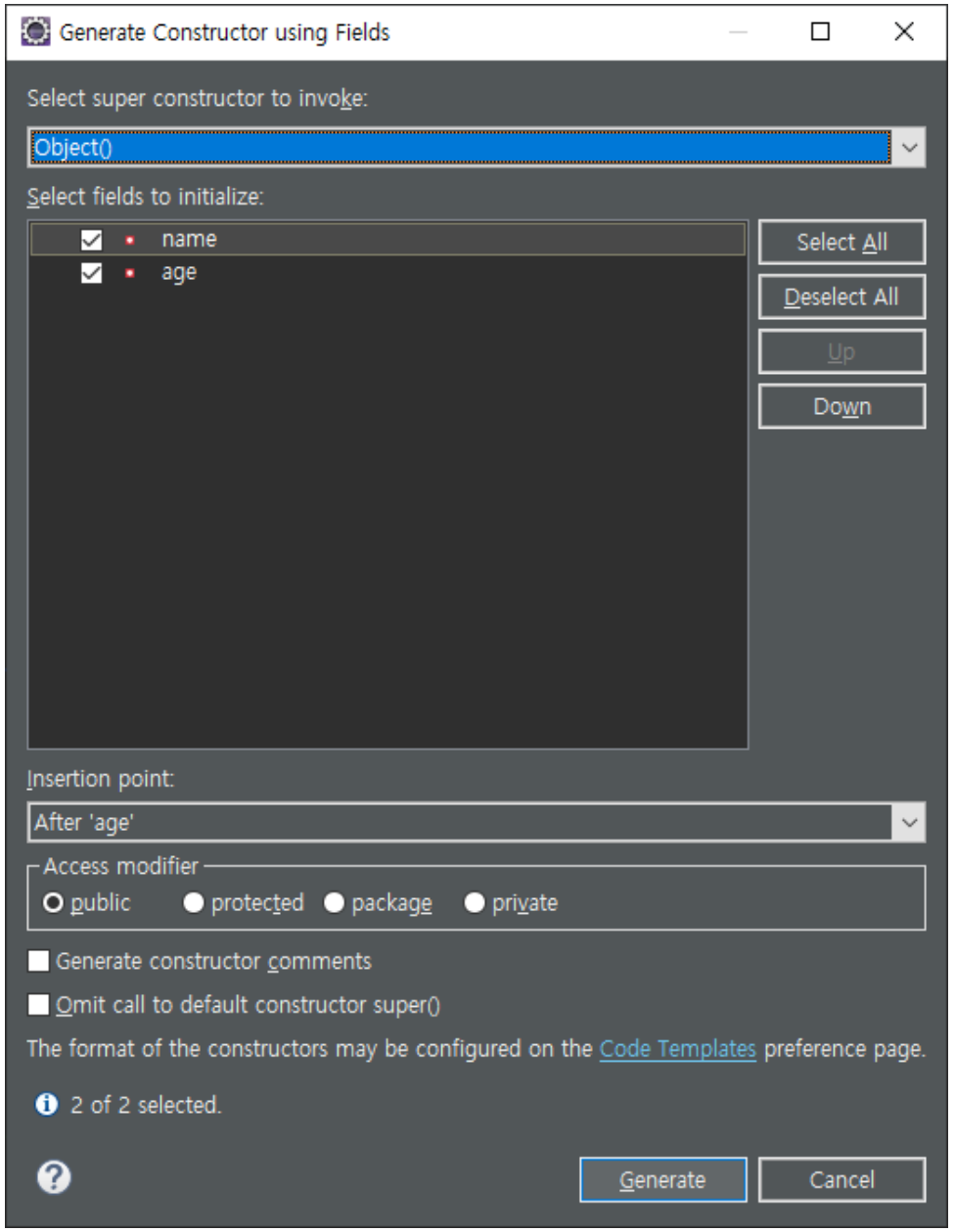
Code Review 3. 오버로딩/오버라이딩

- source -> Generate Constructor using Fields
  - 하고나서 습관적으로 default 생성자 만들기

```java
package edu.medici.overriding;

public class People {
    private String name;
    private int age;

    @Override
    public String toString() {
        return "People [name=" + name + ", age=" + age + "]";
    }

    public People() {}
    public People(String name, int age) {
        super();
        this.name = name;
        this.age = age;
    }
    // name
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    // age
    public int getAge() {
        return age;
    }
```

Default 생성자

생성자 오버로딩

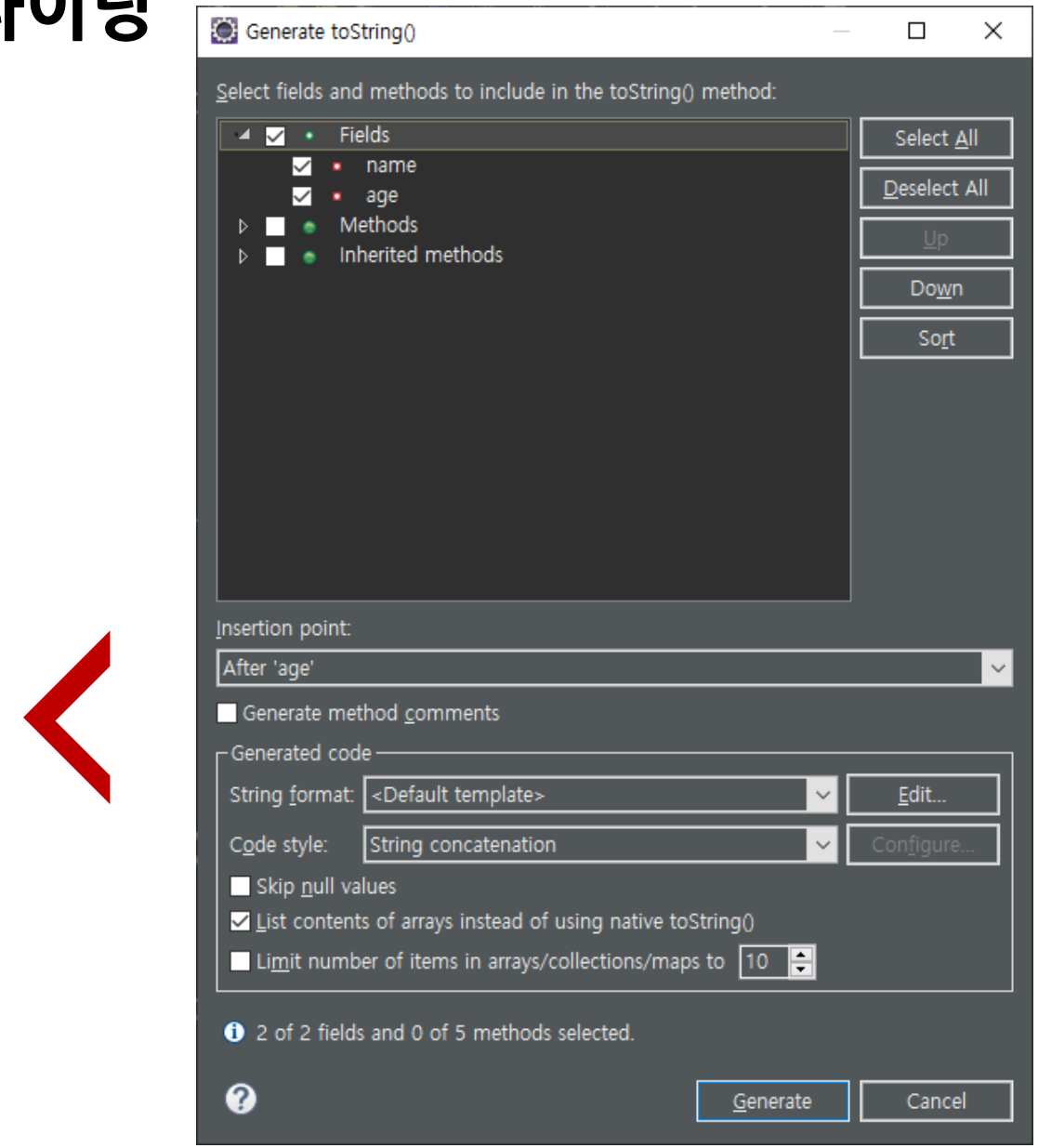


Code Review 3. 오버로딩/오버라이딩

- source -> Generate toString()

```java
package edu.medici.overriding;

public class People {
    private String name;
    private int age;

    @Override
    public String toString() {
        return "People [name=" + name + ", age=" + age + "]";
    }

    public People() {}
    public People(String name, int age) {
        super();
        this.name = name;
        this.age = age;
    }
    // name
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    // age
    public int getAge() {
        return age;
    }
```

오버라이딩



Code Review 3. 오버로딩/오버라이딩

```java
package edu.medici.overriding;

public class Main {

    public static void main(String[] args) {
        People p = new People("사용자", 20);  // toString으로 오버라이딩 했기 때문에 가능
        Student s = new Student("학생명", 14, "202001", "웹프로그래밍");
        Teacher t = new Teacher("한혜진", 20, "202002", "연기");
        // 만들어보기: String tId, String tSubject

//      System.out.println(p);
//      System.out.println(s);
        System.out.println(t);
    }

}
```

```
People [name=사용자, age=20]
People [name=학생명, age=14] Student [sId=202001, subject=웹프로그래밍]
People [name=한혜진, age=20] Teacher [tId=202002, tSubject=연기]
```

```java
@Override
public String toString() {
    return super.toString()+
            " Student [sId=" + sId + ", subject=" + subject + "]";
}

public Student() {} // 생성자 오버로딩 하면 default 생성자 습관적으로 만들기
public Student(String name, int age,
               String sId, String subject) { // 생성자 오버로딩
    super(name, age);
    this.sId = sId;
    this.subject = subject;
}
```

단축키 정리

- `ctrl + F11` : 실행
- `ctrl + N` : 새 파일
  - class, Java Project 이런 것 검색해서 Next
  - 프로젝트 만든 것 하위목록 중 'src'에서 `ctrl + N` 해서 package 검색해서 만들 수도 있음
    - 만들어진 package에 두고 class 만들면 자동으로 package 입력
- `ctrl + M` : 전체화면 & 되돌리기
- `ctrl + space` : 자동완성
  - syso까지만 입력하고 `ctrl+space` 하면 System.out.println();
- `ctrl + /` : 주석처리
- `F3` : println에 커서 놓고 F3 누르면 상세 설명 볼 수 있음, Main에서 메서드에 커서 놓고 F3 누르면 관련된 클래스로 이동
- `ctrl + 1` : 오류 자세히 보기
- `ctrl + s` : 저장(컴파일)
  - IDE에서 중간중간 변경된 사항을 저장하지 않으면 컴파일 되지 않은 상태이므로 오류가 생길 수 있음
- `ctrl + d` : 줄 없애기
- `alt + 위아래방향키` : 해당 라인 위치이동


Thank you