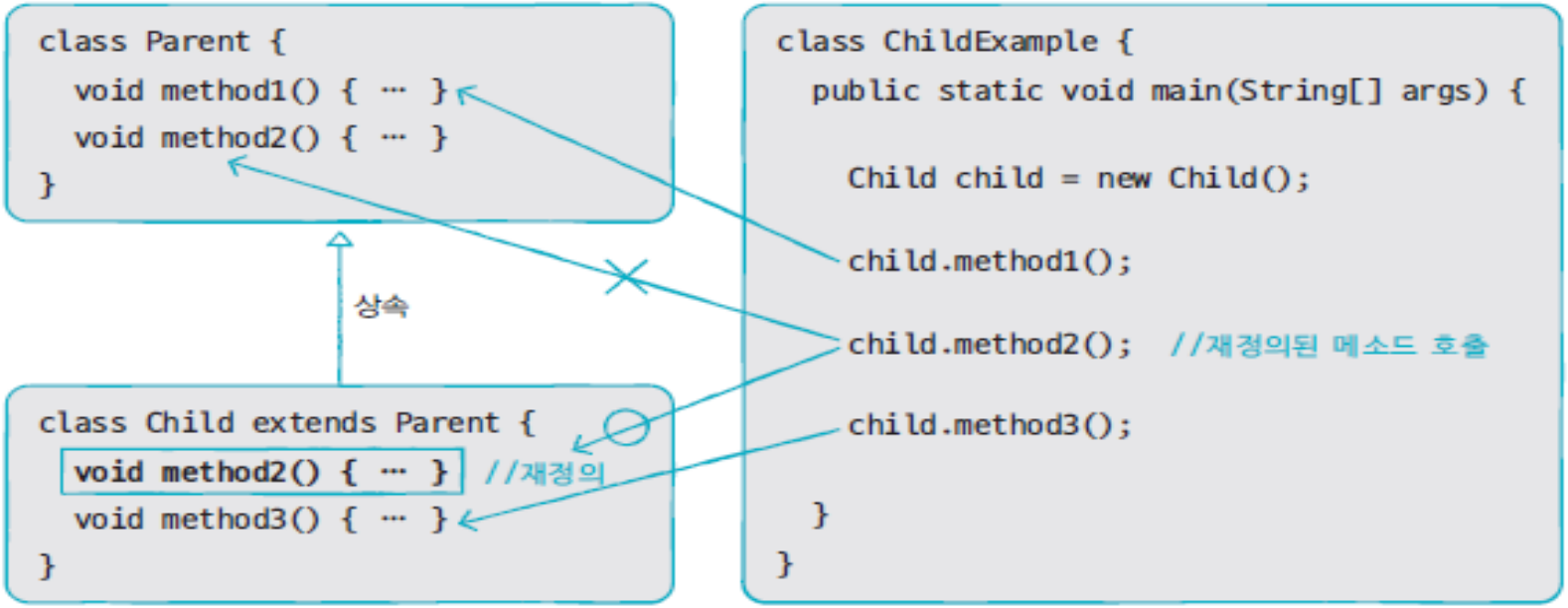
■ 메소드 재정의 (Overriding)

● 부모 클래스로부터 상속받은 메소드를 자식 클래스에 맞게 변경하는 것

● 부모 클래스의 메소드가 숨겨지며, **재정의된** 자식 객체의 메소드가 호출



■ 메소드 재정의 (Overriding) 의 조건

- 선언부가 같아야 한다.(이름, 매개변수, 리턴타입)
- 접근제어자를 좁은 범위로 변경할 수 없다.

  - public ➔ default 또는 private 변경 불가

  - default ➔ public 또는 protected 변경 가능
- 조상클래스의 메서드보다 많은 수의 예외를 선언할 수 없다.

```
class Parent {
    void parentMethod() throws IOException, SQLException {
        // ...
    }
}

class Child extends Parent {
    void parentMethod() throws IOException {
        //..
    }
}

class Child2 extends Parent {
    void parentMethod() throws Exception {
        //..
    }
}
```

■ @Override 애노테이션 사용

- 컴파일러에게 부모 클래스의 메소드 선언부와 동일한지 검사 지시
- 정확한 메소드 재정의를 위해 붙여 주는 것을 권장

ch10.Parent

```
public class Parent {

  public void run() {
    System.out.println("Parent run");
  }

}
```

ch10.Child

```
public class Child extends Parent {

  @Override
  public void run() {
    System.out.println("Overriding");
    System.out.println("Child run");
  }

}
```

■ Overriding

ch10.USB

```
double capacity = 16.0;
double speed = 100.0;

public USB() {
  System.out.println("USB 객체 생성");
}

public void copy() {
  System.out.println(speed + "MB/s 복사");
}

public double getCapacity() { return capacity; }
public void setCapacity(double capacity) {
  this.capacity = capacity;
}

public double getSpeed() { return speed; }
public void setSpeed(double speed) {
  this.speed = speed;
}
```

■ Overriding

ch10.S_USB

```
public class S_USB extends USB {
  double capacity = 0;
  double speed = 0;

  public S_USB() {
    System.out.println("S_USB 객체 생성");
    capacity = 32.0;
    speed = 200.0;
  }

  @Override
  public void copy() {
    System.out.println(speed + "MB/s 복사");
  }

  public void checkCapacity() {
    System.out.println(capacity + "GB");
  }
}
```

■ Overriding

ch10.USBMain

```
public class USBMain {

  public static void main(String[] args) {
    S_USB usb = new S_USB();
    usb.copy();
    usb.checkCapacity();
  }

}
```

```
USB 객체 생성
S_USB 객체 생성
200.0MB/s 복사
32.0GB
```

■ 연습문제 (ch10.연습문제01)

● 결과와 같이 출력될 수 있도록 코드 작성하기

```
public class 연습문제01 {
  public static void main(String[] args) {

    Ramen ramen1 = new SpicyRamen("불 라면");
    String taste1 = ramen1.getTaste();
    System.out.println(taste1);

    Ramen ramen2 = new SaltyRamen("소금 라면");
    String taste2 = ramen2.getTaste();
    System.out.println(taste2);

  }
}
```

```
불 라면 => 매운 라면맛
소금 라면 => 짠 라면맛
```

■ 연습문제 (ch10.연습문제01)

● 결과와 같이 출력될 수 있도록 코드 작성하기

```
public class Ramen {
  String taste;
  String name;

  public String getTaste() {
    return "라면맛";
  }
}
```

```
public class SpicyRamen (  ①  ) {
  public SpicyRamen(String name) {
    super.name = name;
  }

  // ② 오버라이드 코드 작성

}
```

```
public class SaltyRamen (  ③  ) {
  public SaltyRamen(String name) {
    super.name = name;
  }

  // ④ 오버라이드 코드 작성

}
```

■ 오버로딩 (Overloading)

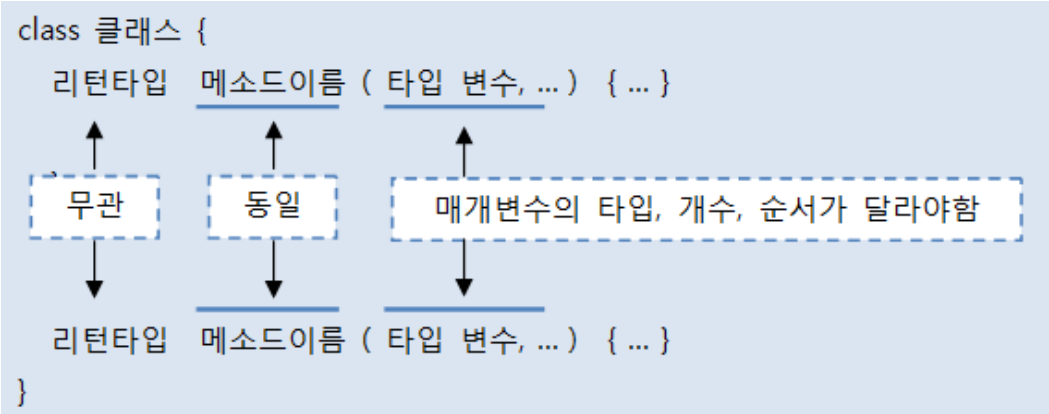
- 클래스 내에 같은 이름의 메소드를 여러 개 선언하는 것
- 하나의 메소드 이름으로 다양한 매개값 받기 위해 메소드 오버로딩
- 오버로딩의 조건: 매개변수의 타입, 개수, 순서가 달라야 됨

■ 오버로딩의 조건

- 메서드의 이름이 같아야 한다.
- 매개변수의 개수 또는 타입이 달라야 한다. (리턴 타입은 상관없음)
- 매개변수는 같고 리턴타입이 다른 경우는 오버로딩이 성립되지 않는다.

  ※ 오버로딩된 메소드는 호출시 전달하는 인자를 통해서 구분

■ 오버로딩의 조건




```
void println() { .. }
void println(boolean x) { .. }
void println(char x) { .. }
void println(char[] x) { .. }
void println(double x) { .. }
void println(float x) { .. }
void println(int x) { .. }
void println(long x) { .. }
void println(Object x) { .. }
void println(String x) { .. }
```

```
int plus(int x, int y) {
    int result = x + y;
    return result;
}
```

plus(10, 20);

```
double plus(double x, double y) {
    double result = x + y;
    return result;
}
```

plus(10.5, 20.3);

```
int x = 10;
double y = 20.3;
plus(x, y);
```

?

```
int divide(int x, int y) { ... }
double divide(int boonja, int boonmo) { ... }
```

■ Overloading

ch10.OverloadExam

```
public class OverloadingExam {
  void change(int num) { }

  void change(char ch) { }

  void change(int num, char ch) { }

  // 반환 타입만 다른 경우 오류, 동일 메소드로 인식
  int change(int num) { }
}
```

■ super / this 사용 - 1 (변수)

ch10.Top

```
public class Top {
  int num = 10;
}
```

ch10.Bottom

```
public class Bottom extends Top {
  int num = 20;

  void show(int num) {
    System.out.println("지역변수 : " + num);
    System.out.println("전역변수 : " + this.num);
    System.out.println("부모의 전역변수 : " + super.num);
  }
}
```

ch10.SuperThis1

```
public class SuperThis1 {
  public static void main(String[] args) {
    Bottom bottom = new Bottom();
    bottom.show(20);
  }
}
```

```
지역변수 : 20
전역변수 : 20
부모의 전역변수 : 10
```

■ super / this 사용 - 2 (메소드)

ch10.Line2D

```
public class Line2D {
  int x = 0;
  int y = 0;

  public String getLocation() {
    return "x : " + x + ", y : " + y;
  }
}
```

ch10.Line3D

```
public class Line3D extends Line2D {
  int z = 0;

  public String getLocation() {
    return super.getLocation() + ", z : " + z;
  }
}
```

ch10.SuperThis2

```
public class SuperThis2 {
  public static void main(String[] args) {
    Line3D line = new Line3D();
    String location = line.getLocation();
    System.out.println(location);
  }
}
```

```
x : 0, y : 0, z : 0
```