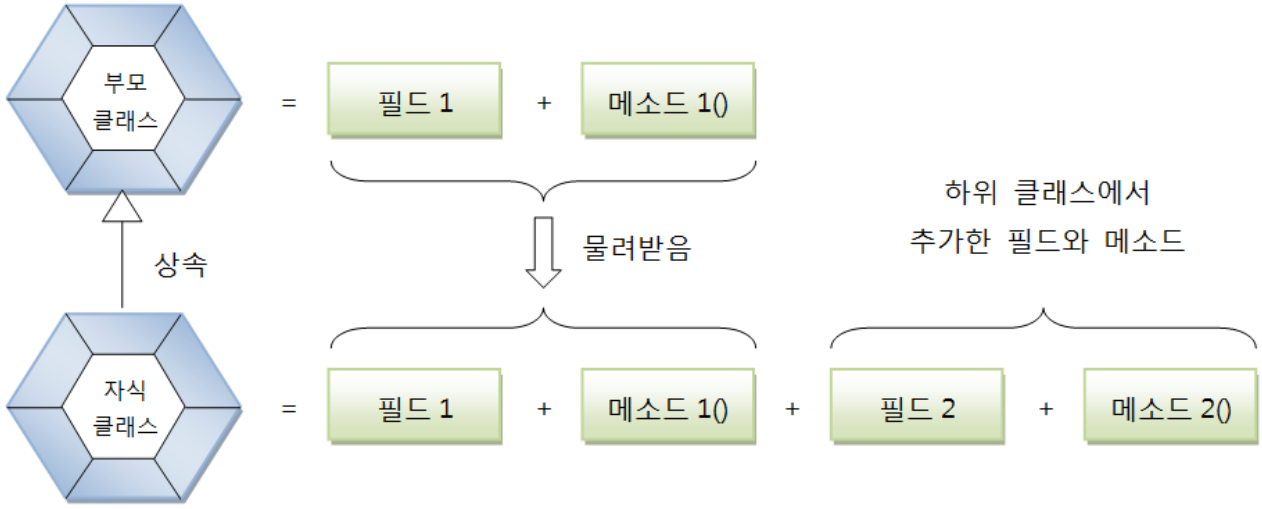
■ 상속(Inheritance)이란?

● 현실 세계

- 부모가 자식에게 물려주는 행위
- 부모가 자식을 선택해서 물려줌

● 객체 지향 프로그램

- 자식(하위, 파생) 클래스가 부모(상위) 클래스의 멤버를 물려받는 것
- 자식이 부모를 선택해 물려받음
- 상속 내용 : 부모의 필드와 메소드



■ 상속(Inheritance) 개념의 활용

● 상속의 효과

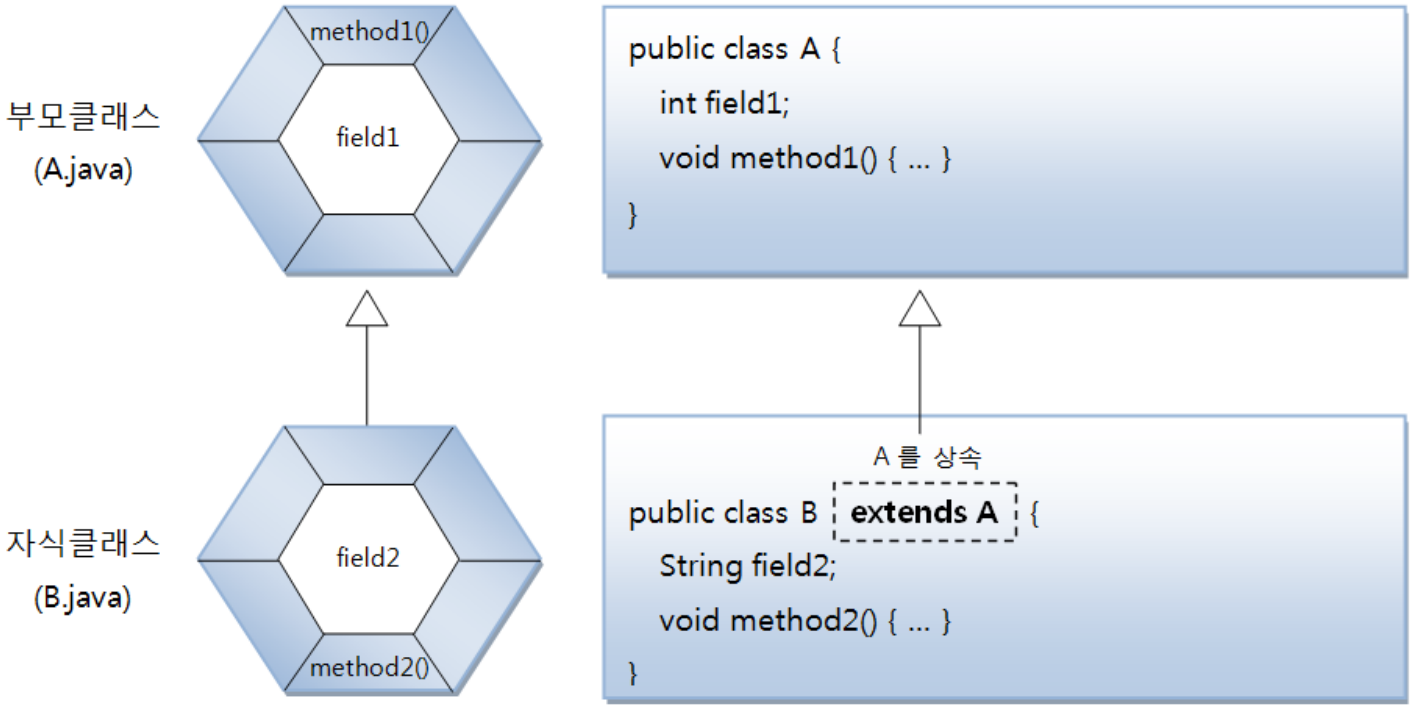
- 부모 클래스 재사용해 자식 클래스 빨리 개발 가능
- 반복된 코드 중복 줄임
- 유지 보수 편리성 제공
- 객체 다형성 구현 가능

● 상속 대상 제한

- 부모 클래스의 private 접근 제어자를 사용한 필드와 메소드
- 부모 클래스가 다른 패키지에 있을 경우,
  default 접근 제어자를 사용한 필드와 메소드 제외

■ extends 키워드

● 자식 클래스가 상속할 부모 클래스를 지정하는 키워드



● 자바는 단일 상속 - 부모 클래스 나열 불가

```
class 자식클래스 extends 부모클래스1, 부모클래스2 {
}
```

■ 상속 (1 / 2)

```
public class Car {
    int tire = 4;
    int door = 2;

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

    void move() {
        System.out.println("이동");
    }

    public int getTire() {
        return tire;
    }

    public void setTire(int tire) {
        this.tire = tire;
    }

    public int getDoor() {
        return door;
    }

    public void setDoor(int door) {
        this.door = door;
    }
}
```

■ 상속 (2 / 2)

```
public class SportsCar extends Car {
    char color = 'R';

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

    void openSunloof() {
        System.out.println("썬루프 열림");
    }
}
```

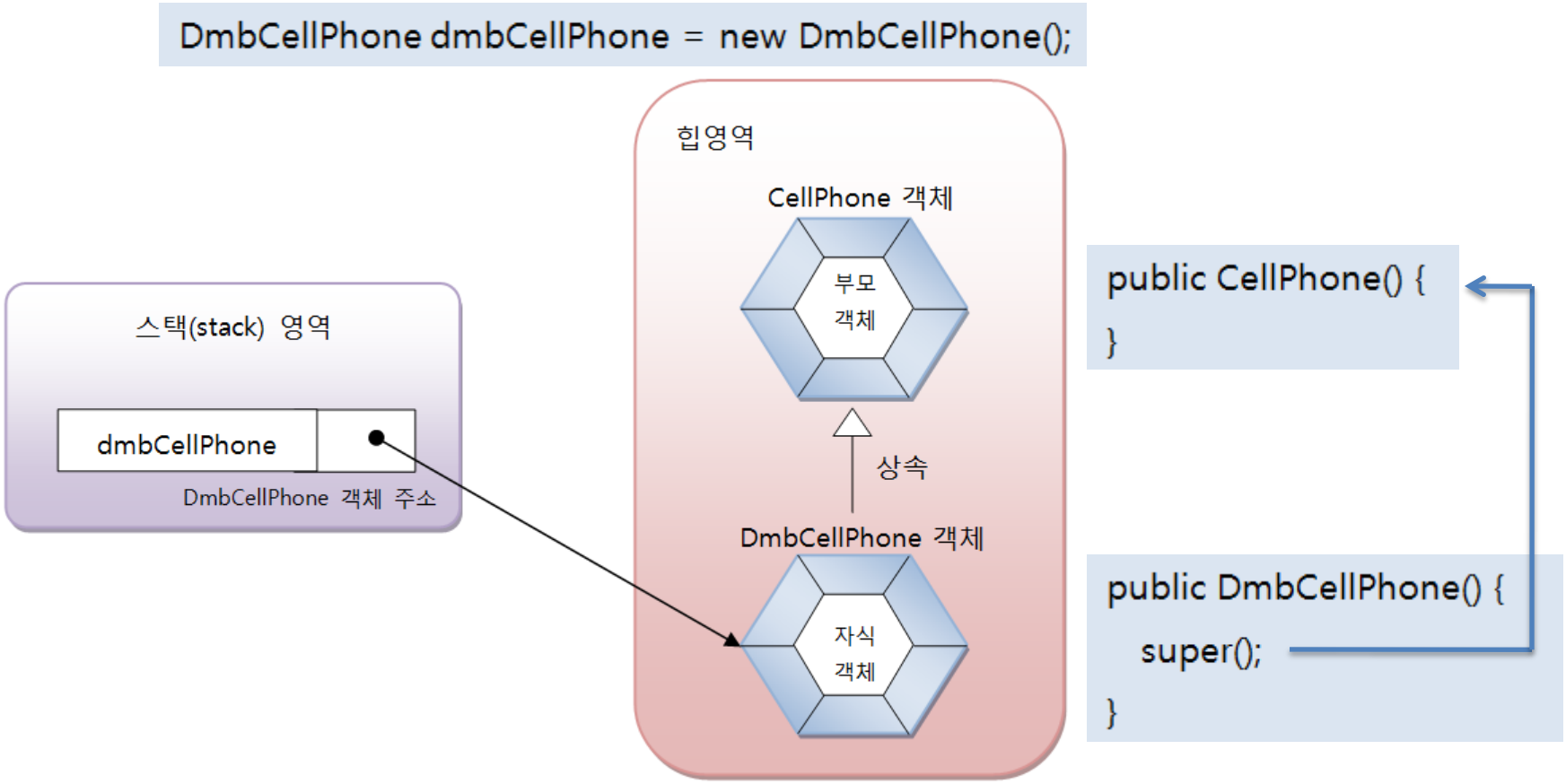
```
public class InheritanceExam1 {

    public static void main(String[] args) {

        SportsCar sc = new SportsCar();

        sc.move();

        sc.openSunloof();

        System.out.println(sc.getDoor());

        System.out.println(sc.getTire());

        System.out.println(sc.toString());

    }

}
```

■ 형변환 (업캐스팅 / 다운캐스팅)

```
public class InheritanceExam2 {
    public static void main(String[] args) {
        SportsCar sc = new SportsCar();
        Car c = null;
        c = (Car) sc; // 업캐스팅
        c.move();

        SportsCar sc2 = (SportsCar) c; // 다운캐스팅
        sc2.openSunloof();

        Car c2 = new Car();
        SportsCar sc3 = null;
        sc3 = (SportsCar) c2; // Car 객체로 생성되었기 때문에 오류

        if(c2 instanceof SportsCar) {
            sc3 = (SportsCar) c2;
            sc3.openSunloof();
        } else {
            System.out.println("캐스팅 불가!");
        }
    }
}
```

■ 부모 생성자 호출 – super()

- 자식 객체 생성할 때는 부모 객체부터 생성 후 자식 객체 생성
- 부모 생성자 호출 완료 후 자식 생성자 호출 완료



■ 명시적인 부모 생성자 호출

● 부모 객체 생성할 때, 부모 생성자 선택해 호출

```
자식클래스( 매개변수선언, ... ) {
   super( 매개값, ... );
   ...
}
```

● super(매개값,...)

- 매개값과 동일한 타입, 개수, 순서 맞는 부모 생성자 호출

● 부모 생성자 없다면 컴파일 오류 발생

● 반드시 자식 생성자의 첫 줄에 위치

● 부모 클래스에 기본(매개변수 없는) 생성자가 없다면 필수 작성

■ 명시적인 부모 생성자 사용 (1 / 2)

```
public class Computer {
    int cpu;

    public Computer() {}

    public Computer(int cpu) {
        this.cpu = cpu;
    }

    public int getCpu() {
        return cpu;
    }

    public void setCpu(int cpu) {
        this.cpu = cpu;
    }
}
```

■ 명시적인 부모 생성자 사용 (2 / 2)

```
public class NoteBook extends Computer {

    public NoteBook() {
        super(3);
    }

    public NoteBook(int cpu) {
        super(cpu);
    }

}
```

■ Object 클래스 – 모든 클래스의 최고조상

```
class MyClass { . . . }
```

```
class MyClass extends Object { . . . }
```

자바 클래스가 아무것도 상속하지 않으면 java.lang 패키지의 Object 클래스를 자동으로 상속한다. 때문에 모든 자바 클래스는 Object 클래스를 직접적 혹은 간접적으로 상속한다.

```
Object obj1=new MyClass();
Object obj2=new int[5];
```

배열도 인스턴스이므로 작성 가능

모든 클래스가 Object 클래스를 직접 혹은 간접적으로 상속하므로, 다음 두 가지가 가능하다.

- ✓ **자바의 모든 인스턴스는 Object 클래스의 참조변수로 참조 가능**
- ✓ **자바의 모든 인스턴스를 대상으로 Object 클래스에 정의된 메소드 호출 가능**

■ Object 클래스 – 모든 클래스의 최고조상

모든 클래스가 Object 클래스를 상속하는 것과 관련해서 기억할 것

- ✓ Object 클래스에는 toString 메소드가 다음의 형태로 정의되어 있다.

  `public String toString( ) { . . . }`

- ✓ 그리고 우리가 흔히 호출하는 println 메소드는 다음과 같이 정의되어 있다.

  `public void println(Object x) { . . . }`

- ✓ 때문에 모든 인스턴스는 println 메소드의 인자로 전달될 수 있다.
- ✓ 인자로 전달되면, toString 메소드가 호출되고, 이 때 반환되는 문자열이 출력된다.
- ✓ 때문에 toString 메소드는 적절한 문자열 정보를 반환하도록 오버라이딩 하는 것이 좋다!

예제 StringToString.java 다시 보기

```
class Friend
{
    String myName;
    public Friend(String name)
    {
        myName=name;
    }
    public String toString()
    {
        return "제 이름은 "+myName+"입니다.";
    }
}
```

```
public static void main(String[] args)
{
    Friend fnd1=new Friend("이종수");
    Friend fnd2=new Friend("현주은");
    System.out.println(fnd1);
    System.out.println(fnd2);
```

실행 결과

```
제 이름은 이종수입니다.
제 이름은 현주은입니다.
```

■ Object 클래스

```
public class Object {
    /* 생략 */

    public native int hashCode();

    public boolean equals(Object arg0) {
        return this == arg0;
    }

    public String toString() {
        return this.getClass().getName() + "@" +
      Integer.toHexString(this.hashCode());
    }

    /* 생략 */
}
```

■ Object 클래스 상속 메소드 사용 – 1

```
public class Book {
    int isbn;
    String title;
    String author;
    int price;

    public int getIsbn() {
        return isbn;
    }
    public void setIsbn(int isbn) {
        this.isbn = isbn;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getAuthor() {
        return author;
    }
    public void setAuthor(String author) {
        this.author = author;
    }
    public int getPrice() {
        return price;
    }
    public void setPrice(int price) {
        this.price = price;
    }
}
```

■ Object 클래스 상속 메소드 사용 – 2

```
public class BookStore {
    Book book;

    public void append(Book book) {
        this.book = book;
    }

    public void remove(Book book) {
        this.book = null;
    }

    public void print() {
        System.out.println(book.toString());
    }
}
```

■ Object 클래스 상속 메소드 사용 – 3

```
public class BookMain {
    public static void main(String[] args) {
        Book book = new Book();

        BookStore store = new BookStore();
        store.append(book);
        store.print();
    }
}
```