**자바 손코딩 준비 정리 - 핵심 파트별 요약**

**▶ 6장: 클래스 개념**

* **클래스 구성 요소**: 필드(Field), 메소드(Method), 생성자(Constructor)
* **this 정보**: 현재 객체를 가리키는 참조
* **접근 제한자**: public, private, protected
* **getter/setter**: private 필드에 접근하는 public 메서드
* **생성자 오버로딩**: 매개변수 개수에 따라 여러 생성자 작성

📌 **예시 코드**

class Person {

private String name;

public Person(String name) {

this.name = name;

}

public String getName() { return name; }

public void setName(String name) { this.name = name; }

}

**▶ 7장: 상속 (Inheritance)**

* **extends**: 자식 클래스가 부모 클래스 상속
* **super**: 부모 생성자 또는 멤버 호출
* **super()**: 부모 클래스의 생성자를 명시적으로 호출
* **오버라이딩 (Overriding)**: 부모 메서드를 자식이 재정의
* **업캐스팅/다운캐스팅**: 부모 타입으로 참조, 필요 시 자식으로 변환

📌 **예시 코드**

class Animal {

public Animal() {

System.out.println("Animal 생성자 호출");

}

public void sound() { System.out.println("소리 없음"); }

}

class Dog extends Animal {

public Dog() {

super(); // 부모 생성자 호출

System.out.println("Dog 생성자 호출");

}

@Override

public void sound() { System.out.println("멍멍"); }

}

Public class AnimalTest{

Public static void main(String[ ] args) {  
 Animal a = new Dog(); // 업캐스팅

If(a instanceof Dog) {

((Dog)a).sound(); 다운캐스팅 후 호출

}

}

}

**▶ 8장: 인터페이스**

* **interface**: 다중 구현 가능, 추상 메서드만 포함
* **implements**: 클래스가 인터페이스를 구현할 때 사용
* **@Override** 필수

📌 **예시 코드**

interface RemoteControl {

void turnOn();

}

class TV implements RemoteControl {

public void turnOn() {

System.out.println("TV On");

}

}

**▶ 9장: 추상 클래스 (abstract class)**

* **abstract class**: 추상 메서드를 포함할 수 있으며 객체 생성 불가
* **abstract method**: 선언만 하고 구현은 하지 않음 → 자식 클래스에서 반드시 오버라이딩
* **일반 메서드와 필드도 함께 가질 수 있음**

📌 **예시 코드**

abstract class Shape {

int x, y;

public void move(int x, int y) {

this.x = x; this.y = y;

}

public abstract void draw();

}

class Circle extends Shape {

@Override

public void draw() {

System.out.println("원을 그립니다.");

}

}

**▶ 참고 예제 상황별 코드**

📌 **업/다운 캐스팅**

Point2D p = new Point3D(); // 업캐스팅

((Point3D)p).setZ(30); // 다운캐스팅 후 자식 필드 접근

📌 **오버라이딩 구조**

class Parent {

public void greet() { System.out.println("Hello from parent"); }

}

class Child extends Parent {

@Override

public void greet() { System.out.println("Hello from child"); }

}

📌 **인터페이스 구현 + 다형성**

interface SoundMaker {

void makeSound();

}

class Cat implements SoundMaker {

public void makeSound() { System.out.println("야옹"); }

}

SoundMaker s = new Cat();

s.makeSound(); // 다형성 기반 호출

✅ **시험 전 암기보다는 익숙해져야 할 코드 패턴**

* 클래스 정의 + 생성자 + this
* 상속 + 오버라이딩 + super()
* 인터페이스 정의 + implements + 다형성
* 추상 클래스 사용 구조 + 오버라이딩
* 업/다운 캐스팅과 instanceof 활용
* 메서드 호출 흐름 (예: ab.f(b) 호출 시 어떤 클래스 메서드가 동작하는가)

위의 코드들은 모두 손으로 직접 써보고 출력 결과를 예상해보는 연습이 매우 중요합니다.