# 디지털 컨버전스 UXUI
# Pront 전문 개발자 양성과정


이상훈
GCCCOMPIL3R@GMAIL.COM
김도혜
KIMDOHYE0728@GMAIL.COM

```java
class ConsTest {
    /* 데이터 저장 영역 시작 */
    int age;
    String name;
    /* 데이터 저장 영역 끝 */

    /* 기능 설정 영역 시작 */
    // 생성자는 아래와 같이 여러 형식으로 구성할 수 있다!
    // 이와 같이 이름이 같고 입력이 다른 케이스로 매서드를 사용하는 방식에 대해 함수 오버로딩이라고 한다.
    // 입력의 개수를 가지고 판단하지 않으며 사용되는 입력에 데이터타입을 보고 판단한다는 점에 주의해야 한다.
    ConsTest() {
        System.out.println("안녕 나는 ConsTest() 이라고해!");
    }
    ConsTest(int a) {
        System.out.println("안녕 나는 ConsTest(int a) 이라고해!");

        age = a;
    }
    ConsTest(float f) {
        System.out.println("안녕 나는 ConsTest(float f) 라고해!");
    }
    ConsTest(int a, String n) {
        System.out.println("안녕 나는 ConsTest(int a, String n) 이라고해!");
```

# 1. 생성자 특징

1. 리턴타입이 없다.

2. 클래스 이름과 매서드 이름이 같다.

3. new를 할 때 호출된다.

4. 생성자는 여러 개로 만들 수 있다. 이걸 함수 오버로딩이라고 한다.

5. 각각의 생성자는 독립적이고 영향을 주지 않는다.

# 2. 객체

객체는 "데이터가 메모리에 올라가는 것"

이라고 생각하면 된다.

추상화 시킨 것.

operator의 뜻이 운전을 하는 사람,

사업을 하는 사람이라고 나오는데

아마 메모리에 올리는 역할을 해서

이런 단어를 쓴 게 아닐까?

```
public class Constructer {
    public static void main(String[] args) {
        ConsTest ct1 = new ConsTest();
        ConsTest ct2 = new ConsTest( a: 10);
        ConsTest ct3 = new ConsTest( a: 20,  n: "hi");
        ConsTest ct4 = new ConsTest( a: 40);
        ConsTest ct5 = new ConsTest( f: 3.3f);

        // * 우리가 사용하는 모든 데이터는
        // 메모리(PC상에서 DRAM)에 올라가야지만 사용할 수 있고 눈으로 볼
        // 결국 객체라는 단어 자체는 메모리에 데이터를 올렸습니다의 추상화

        // 원래 여기서 사용했던 Setter는 어디로 갔나요 ?
        // Setter가 없는데도 결과가 나오네요 ?
        // 결국 생성자는 객체를 처음 생성할 때 초기값을 설정해주는 역할을
        // (결론적으로 초기 생성에 한정하여 Setter의 역할을 대신해줄 수

        /*
        ct2.setName();
        ct2.setAge();
        ct2.setMajor();
              vs
        new ConsTest(이름, 나이, 전공);
         */
```

```java
class TestDice {
    int comDice;
    int userDice;

    TestDice() {
        comDice = getRandomDice();
        userDice = getRandomDice();
    }
    int getRandomDice() {
        return (int)(Math.random() * 6 + 1);
    }
    Boolean userWin() {
        System.out.printf("%d(컴퓨터) vs %d(사용자)\n", comDice, userDice);

        if (comDice > userDice) {
            return false;
        } else if (comDice < userDice) {
            return true;
        } else {
            System.out.println("무승부입니다.");
            return false;
        }
    }
}
```

```java
public class Prob34 {
    public static void main(String[] args) {
        // 우리가 이전에 Random과 제어문을 활용해서 주사위 게임을 만들었던적이 있다.
        // 이 주사위 게임을 class 방식으로 다시 만들어보자!
        // 컴퓨터도 주사위를 굴리고 사용자도 주사위를 굴려서 누가 더 큰 숫자를 얻었는지 확인해보자!
        TestDice td = new TestDice();

        if (td.userWin()) {
            System.out.println("사용자가 승리하였습니다.");
        } else {
            System.out.println("컴퓨터도 못이겼다. 폐관수련이 답이다.");
        }
    }
}
```

## 3. 클래스를 이용해서 주사위 던지기 게임 만들기

클래스를 이용해 메서드를 만들어 논다면
main에서 간단한 코드로 출력을 할 수 있다!

# 4. 클래스 내부에 여러 개의 매서드 만들기

main에서 코드를 복잡하게 사용하는 것을 막을 수 있다!

```
class TestDice3 {
    int comDice;
    int userDice;

    Scanner scan;

    TestDice3() {
        comDice = getRandomDice();
        userDice = getRandomDice();

        scan = new Scanner(System.in);
    }

    int getRandomDice() {
        return (int)(Math.random() * 6 + 1);
    }

    void checkWinner() {
        switch (whoWin()) {
            case 1:
                System.out.printf("폐관수련입니다. %d(컴퓨터) vs %d(사용자)\n", comDice, userDice);
                break;
            case 2:
                System.out.printf("사용자가 이겼습니다. %d(컴퓨터) vs %d(사용자)\n", comDice, userDice);
                break;
            case 3:
                System.out.printf("비겼으니 형은 면하였습니다. %d(컴퓨터) vs %d(사용자)\n", comDice, userDice);
                break;
        }
    }
```

```
int whoWin() {
    if (comDice > userDice) {
        return 1;
    } else if (comDice < userDice) {
        return 2;
    } else {
        return 3;
    }
}

Boolean redoDiceGame() {
    System.out.print("게임을 계속 하시겠습니까 ? 0(아니오), 1(예) ");

    int num = scan.nextInt();

    Boolean isTrue = false;

    switch (num) {
        case 0:
            isTrue = false;
            break;
        case 1:
            // 게임을 다시 재개하므로 주사위값을 새롭게 설정할 필요가 있다.
            comDice = getRandomDice();
            userDice = getRandomDice();
            isTrue = true;
            break;
    }

    return isTrue;
```