Java 치트시트

1. 기본 구조 및 컴파일

Java 프로그램은 클래스(class) 안에 `main` 메서드를 포함하는 구조를 가집니다.

```
// HelloWorld.java
public class HelloWorld {
    // 프로그램 시작점
    public static void main(String[]
args) {
        System.out.println("Hello,
World!");
    }
}
```

- 컴파일 및 실행:

```
javac HelloWorld.java
java HelloWorld
```

2. 변수, 타입, 연산자

- 기본 타입 (Primitive Types):
  - 정수: `byte`, `short`, `int`, `long`
  - 실수: `float`, `double`
  - 문자: `char`
  - 불리언: `boolean`
- 참조 타입 (Reference Types): `String`, `Array`, `Class` 등.
- 변수 선언: `type variableName = value;` (예: `int age = 30;`)
- 상수: `final double PI = 3.14;`
- 타입 캐스팅: `(type)value` (예: `(int) 3.14`)
- 연산자: C++과 유사 (산술, 관계, 논리).

3. 제어 흐름

- `if-else`:

```
if (condition) {
    // ...
} else if (condition) {
    // ...
} else {
    // ...
}
```

- `switch`:

```
switch (day) {
    case "MONDAY":
        // ...
        break;
    case "TUESDAY":
        // ...
        break;
    default:
        // ...
}
```

- `for` 루프:

```
for (int i = 0; i < 5; i++) {
    // ...
}
```

- 향상된 `for` 루프 (Enhanced for-loop):

```
int[] numbers = {1, 2, 3};
for (int number : numbers) {
    System.out.println(number);
}
```

- `while` 루프: `while (condition) { ... }`
- `do-while` 루프: `do { ... } while (condition);`

4. 클래스와 객체

- 클래스 정의:

```
public class Car {
    // 필드 (멤버 변수)
    private String color;
    private int speed;

    // 생성자
    public Car(String color) {
        this.color = color;
        this.speed = 0;
    }

    // 메서드
    public void accelerate(int amount)
{
        this.speed += amount;
    }
}
```

- 객체 생성: `Car myCar = new Car("blue");`
- 멤버 접근: `myCar.accelerate(10);`

5. 상속과 인터페이스

- 상속 (`extends`): 클래스의 필드와 메서드를 재사용.

```
public class ElectricCar extends Car {
    // ... 추가 필드 및 메서드
}
```

- 인터페이스 (`interface`): 클래스가 구현해야 할 메서드의 명세.

```
public interface Drivable {
    void drive();
}

public class MyCar implements Drivable
{
    @Override
    public void drive() {
        // ... 구현
    }
}
```

6. 컬렉션 프레임워크

`java.util` 패키지에 포함.

ArrayList

크기가 동적으로 변하는 배열.

```
import java.util.ArrayList;
ArrayList<String> names = new
ArrayList<>();
names.add("Alice");
names.add("Bob");
String first = names.get(0);
```

HashMap

키-값 쌍을 저장.

```
import java.util.HashMap;
HashMap<String, Integer> ages = new
HashMap<>();
ages.put("Alice", 30);
ages.put("Bob", 25);
int aliceAge = ages.get("Alice");
```

7. 예외 처리

- `try-catch-finally`:

```
try {
    // 예외가 발생할 수 있는 코드
} catch (ExceptionType e) {
    // 예외 처리
} finally {
    // 항상 실행되는 코드 (자원 해제 등)
}
```

- `throws`: 메서드가 특정 예외를 발생시킬 수 있음을 선언.

```
public void readFile() throws
java.io.IOException {
    // ...
}
```

8. 스트림 API (Java 8+)

컬렉션 데이터를 함수형 스타일로 처리.

```
import java.util.List;
import java.util.stream.Collectors;

List<String> names = List.of("Alice",
"Bob", "Charlie");
List<String> upperCaseNames =
names.stream()
                        .map(String::toUpperCase)
                            .filter(name
→ name.startsWith("A"))
                        .collect(Collectors.toList(
```