Python vs Java — Static, Class Properties, Class Methods, Overriding Full Detailed Notes (Textbook Style)

Introduction

This note gathers the complete discussion, your questions, and the full detailed explanations about Python class properties, static methods, class methods, and how they compare with Java's static variables and static methods. It also includes an in-depth explanation of why Java static methods do NOT support overriding while instance methods DO.

Python: Instance, Class, and Static Members

Python supports three types of methods and two kinds of properties.

1. Instance Property & Instance Method
- Belongs to the object.
- Each object has its own data.
- Uses `self`.

Example:
```
class A:
def __init__(self):
self.x = 10

def show(self):
print(self.x)
```

2. Class Property & Class Method
- Shared across all objects.
- Defined directly inside the class.
- Class methods use `@classmethod` and receive `cls`.
- Can modify class-level state.
- **Supports polymorphism** because `cls` refers to the actual class at runtime.

Example:
```
class A:
count = 0

@classmethod
```

def inc(cls):
cls.count += 1
```

### 3. Static Method
- Uses `@staticmethod`.
- Receives no `self` or `cls`.
- Behaves like a normal function inside a class.
- Not polymorphic.

Example:
```
class A:
@staticmethod
def add(a, b):
return a + b
```

## Java: Instance vs Static

Java supports instance members and static members.

### 1. Instance Members
- Belong to objects.
- Can be overridden.
- Follow runtime polymorphism (dynamic dispatch).

### 2. Static Members
- Belong to the class itself.
- Static methods cannot be overridden—only hidden.
- Static variables belong only to the class, not objects.
- Resolved at compile time (static binding).

Example:
```
class A {
static int count = 0;
static void inc() { count++; }
}
```

## Are Java static methods the same as Python class methods?

### Short Answer: **NO**

Java static method = Python @staticmethod
```

Java has NO equivalent to Python @classmethod

Why?
- Java static methods do not receive class reference.
- Python classmethods receive `cls`, the class object.
- Python classmethod supports polymorphism; Java static does not.

But both can access class-level properties
Your observation was correct:
- Java static methods can use static variables.
- Python classmethods can use class variables.

But the mechanism is different.

Why Python classmethod ≠ Java static

Key Differences

Feature	Python classmethod	Java static
Receives class (cls)?	Yes	No
Polymorphic?	Yes	No
Overridable?	Yes	No
Bound to	Class (dynamic)	Class (fixed)
Equivalent?	No	Python staticmethod

Your Question: Why Java static methods don't override, but instance methods do?

This is one of the most frequently misunderstood parts of Java.

The Core Reason:
Static methods belong to the CLASS, not the OBJECT.

Overriding applies only to object-based methods.
Overriding requires:
- Dynamic dispatch
- Runtime decision
- Based on actual object type

Static methods:
- Resolved at compile time
- Based on reference type
- Not part of object memory
- Do not participate in runtime polymorphism

Example You Provided

```
class A {
static int count = 0;
static void inc() { count++; }
}

class B extends A {
static int count = 100;
}
```

Calling:
```
B.inc();
System.out.println(B.count);
```

Produces:
- `B.inc()` calls **A.inc()**, because static methods are not overridden.
- A.inc() increments **A.count**, not B.count.
- B.count stays 100.

Why Java Designers Did This
If static methods were dynamically dispatched:
- Their behavior would depend on objects.
- But static methods are not supposed to depend on objects.
- This would break the meaning of "static".

Final Reason (Summary)
- Instance methods → runtime binding → overriding allowed.
- Static methods → compile-time binding → overriding NOT allowed → only hiding.

Final Memory Tricks

- Static = belongs to class = no overriding.
- Instance = belongs to object = overriding works.
- Java static == Python staticmethod.
- Python classmethod has no Java equivalent.
- If a method needs polymorphism → instance or classmethod (Python).
- If a method must not depend on objects → static (both languages).