final int vs static final int in Java

Key Differences

- **final int**: Constant value per instance (instance-level constant)
- **static final int**: Constant value per class (class-level constant)

Example (final int - Instance Constants)

```
public class BankAccount {
    // final int - constant per instance
    private final int accountNumber;
    private final String accountHolder;
    private double balance;

    // Constructor sets final values
    public BankAccount(int accNumber,
                       String holder,
```

Example (static final int - Class Constants)

```
public class MathConstants {
    // static final int - shared by all
        instances
    public static final int MAX_ITERATIONS =
         1000;
    public static final int
        DEFAULT_PRECISION = 10;
    public static final int
```

Practical Use Cases and Examples

final int - Object-Specific Constants

```
public class Employee {
    // Each employee has their own
    // immutable ID and hire year
    private final int employeeId;
    private final int hireYear;
    private String name;
    private double salary;

    public Employee(int id, int year,
                    String name, double salary
                        ) {
        this.employeeId = id;
        this.hireYear = year;
```

static final int - Shared Constants

```
public class Configuration {
    // Application-wide constants
    public static final int MAX_USERS =
        1000;
    public static final int SESSION_TIMEOUT
        = 1800;
    public static final int MAX_FILE_SIZE =
        10485760; // 10MB

    // Status codes
    public static final int STATUS_ACTIVE =
        1;
    public static final int STATUS_INACTIVE
```

Real-World Examples

Example (final int - Immutable Object State)

```
public class Transaction {
    // Each transaction has immutable ID
    private final int transactionId;
    private final long timestamp;
    private final double amount;
    private String status;

    public Transaction(int id, double amount
        ) {
        this.transactionId = id;
        this.amount = amount;
        this.timestamp = System.
```

Example (static final int - Application Constants)

```
public class HttpStatusCodes {
    // HTTP status codes - shared constants
    public static final int HTTP_OK = 200;
    public static final int HTTP_CREATED =
        201;
    public static final int HTTP_BAD_REQUEST
         = 400;
    public static final int
        HTTP_UNAUTHORIZED = 401;
    public static final int HTTP_FORBIDDEN =
         403;
    public static final int HTTP_NOT_FOUND =
```

When to Use Each

Use final int when:

- Each instance needs its own constant value
- Value is determined at object creation
- Value differs between instances
- Examples:
  - Employee ID
  - Account number
  - Transaction ID
  - Student roll number
  - Order ID

Use static final int when:

- Constant value is shared across all instances
- Value is known at compile time
- Value is the same for all objects
- Examples:
  - Mathematical constants
  - Configuration values
  - Status codes
  - Application limits
  - Default settings

Example (final int Characteristics)

```
// Memory allocated per instance
```

Example (static final int Characteristics)

Summary: final int vs static final int

final int	**static final int**
Instance-level constant	Class-level constant
Each object has its own copy	Single copy shared by all objects
Memory allocated per instance	Memory allocated once per class
Value can differ between instances	Same value for all instances
Set in constructor	Usually set at declaration
Use for:	**Use for:**
Object-specific identifiers	Application-wide constants
Immutable instance properties	Mathematical constants
Runtime-determined constants	Configuration values
Unique per-object values	Shared settings

Key Point

- `final int`: "This object has a value that won't change after creation"
- `static final int`: "All objects of this class share this unchanging value"