Java 内存管理机制

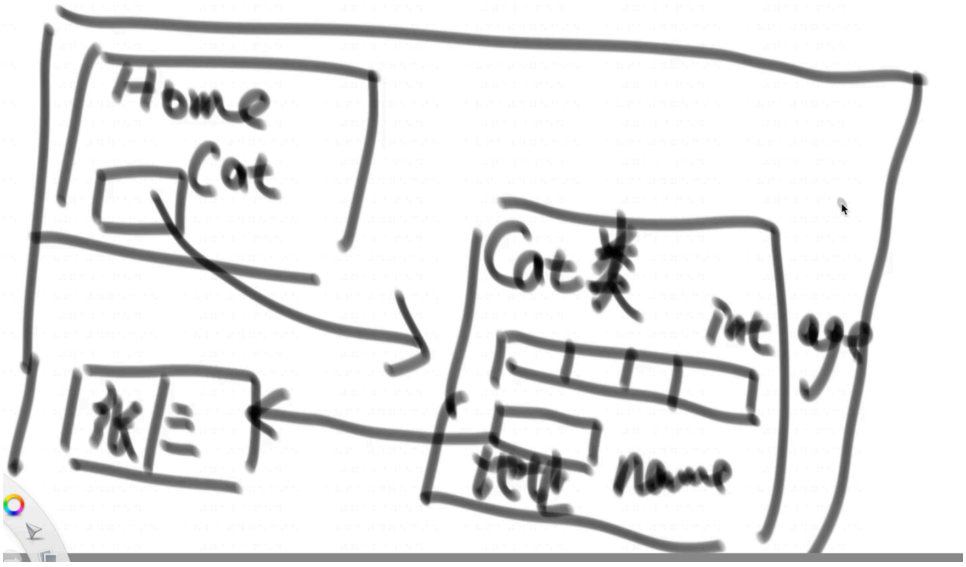
```
public class Cat {
    int age;
    String name;

    public Cat(int age, String name) {
        this.age = age;
        this.name = name;
    }
}
```

```
public class Home {
    Cat cat;

    public Home(Cat cat) {
        this.cat = cat;
    }

    public static void main(String[] args) {
        new Home(new Cat(1, "张三"));
    }
}
```



🧠 Java 内存中发生的事情

- 当执行 `new Home(new Cat(1, "张三"))` 时，JVM 会先在 **堆（heap）中** 分别创建两个对象：一个 `Cat` 对象和一个 `Home` 对象
- 由于 `Home` 类中的 `Cat cat` 成员变量不是 `static`，它属于 `Home` 实例，因此在创建 `Home` 对象时，内存中会同时分配一块区域来保存对 `Cat` 对象的**引用**（即地址）
- `new Cat(1, "张三")` 会在堆中创建一个 `Cat` 类的对象
- Cat类的这个对象中包含两个字段：`int age` 是基本类型，直接以数值形式保存在对象内部；`String name` 是引用类型，保存的是一个指向 `"张三"` 字符串对象的地址
- 虽然 `"张三"` 是字符串字面量，它最初会被放入方法区的字符串常量池（String Pool），但在现代 JVM 中，对应的 `String` 实例常常也会在堆中存在
- 所以，`name` 实际上指向一个 `String` 对象，这个对象本身也在堆中，并包含一个底层的 `char[]` 字符数组，其内容是字符序列 `"张三"`
- 至此，我们在堆内存中形成了一个链式结构：`Home` → `Cat` → `String("张三")` → `char[]`
- 在栈（stack）内存中，当 `main` 方法运行时，JVM 会为其创建一个栈帧，临时保存变量引用
- `main` 方法中的 `new Home(...)` 生成的 `Home` 对象引用会被保存在栈中一个变量中（例如变量名为 `home`，尽管这里没有定义变量，也可能是匿名调用），该引用指向堆中的 `Home` 对象

- 总结来说，整个执行过程中，所有对象都存在于堆内存，通过一层层的引用地址连接起来
- 而临时变量（如 `home`）的引用则存在于栈内存，起到连接栈与堆的作用
- 这构成了一张完整的对象引用图，是理解 Java 内存管理机制的基础