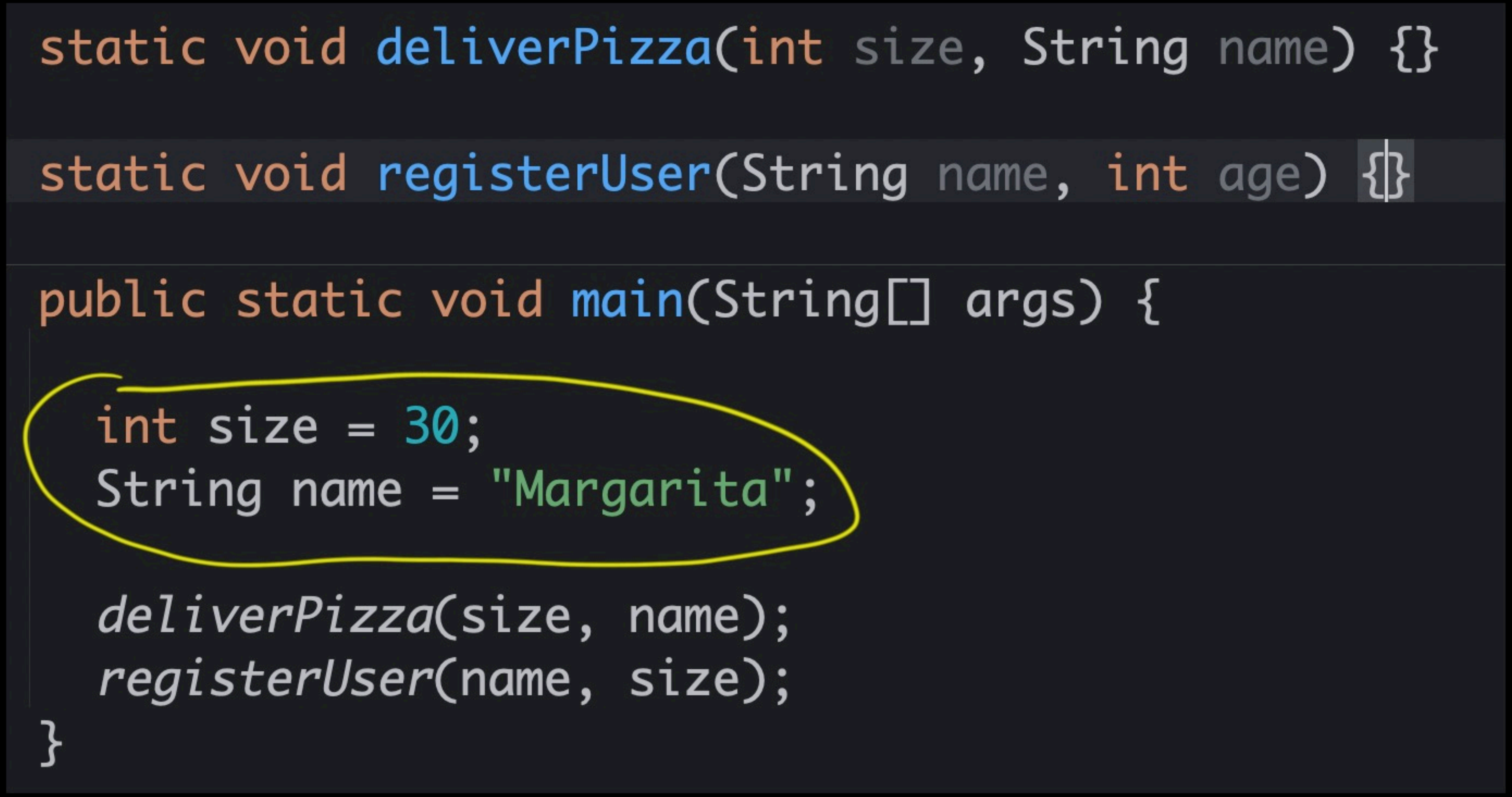
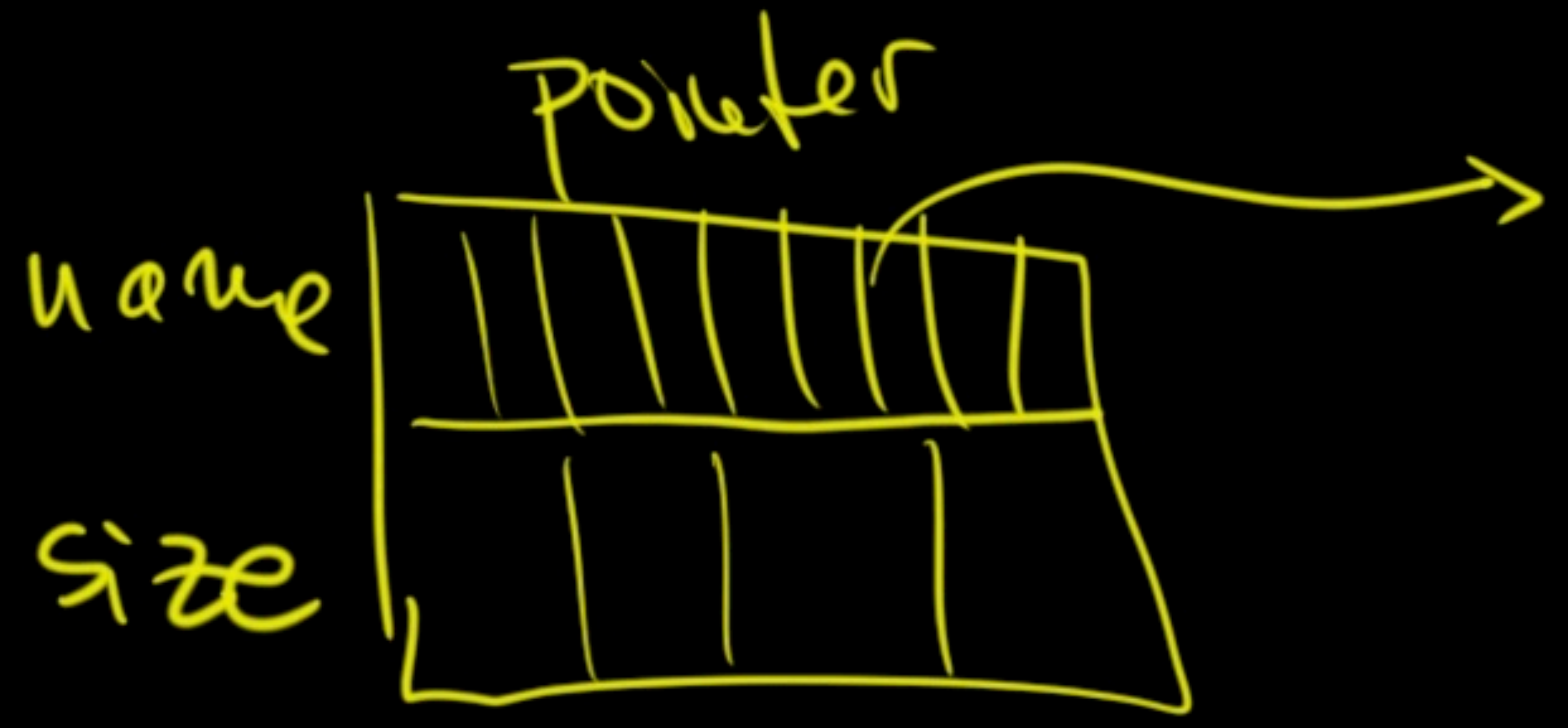


```
static void deliverPizza(int size, String name) {}

static void registerUser(String name, int age) {}

public static void main(String[] args) {

  int size = 30;
  String name = "Margarita";

  deliverPizza(size, name);
  registerUser(name, size);
}
```

```java
Pizza pizza = new Pizza();
```

```java
public class Pizza {

  String name;
  int size;

  public Pizza() {}

}
```

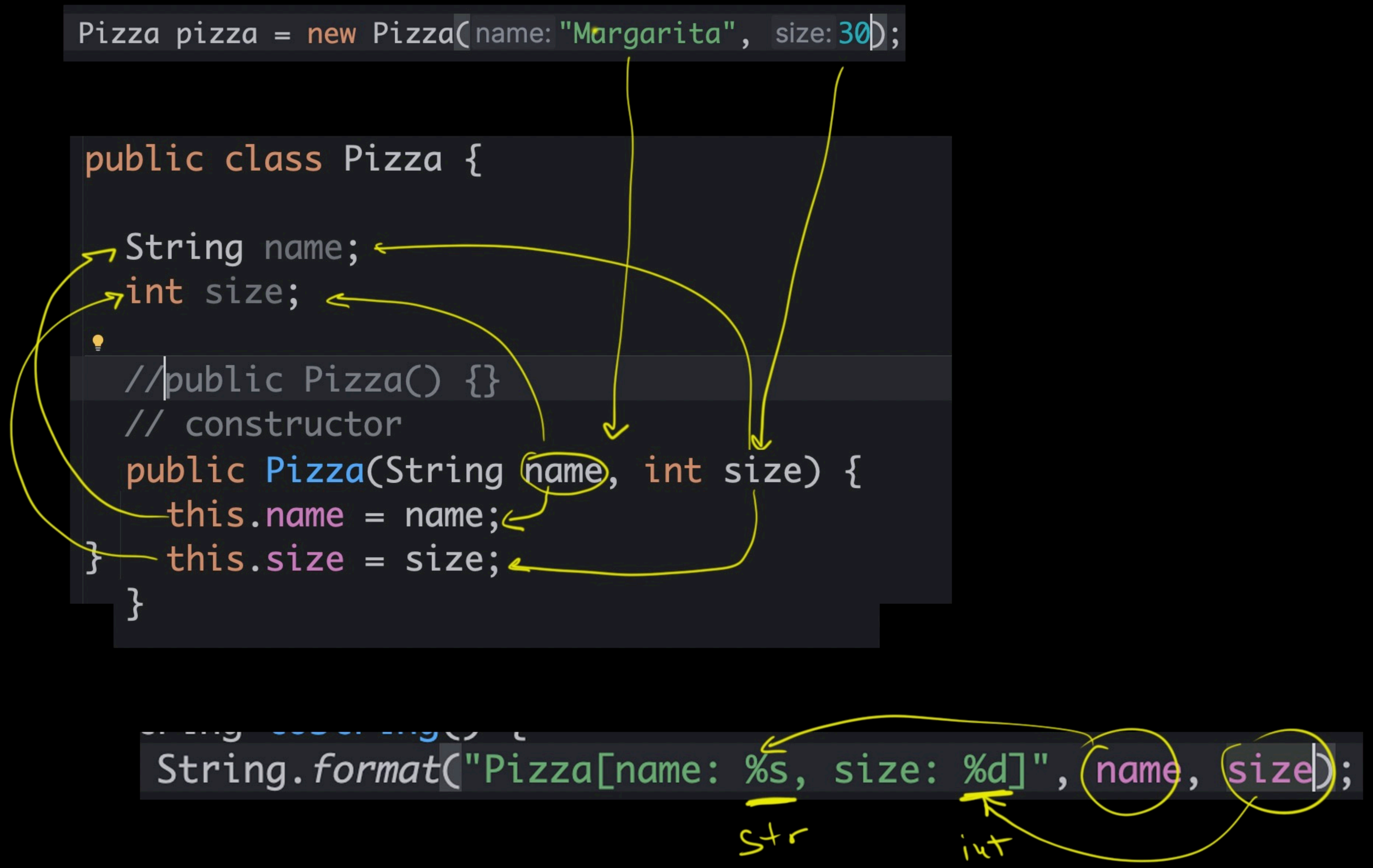


$$\underleftarrow{\text{int add(int,int)}}$$

```java
// constructor
public Pizza(String name, int size)
```

$$\underrightarrow{\text{(name , size)} \rightarrow \text{Pizza}}$$

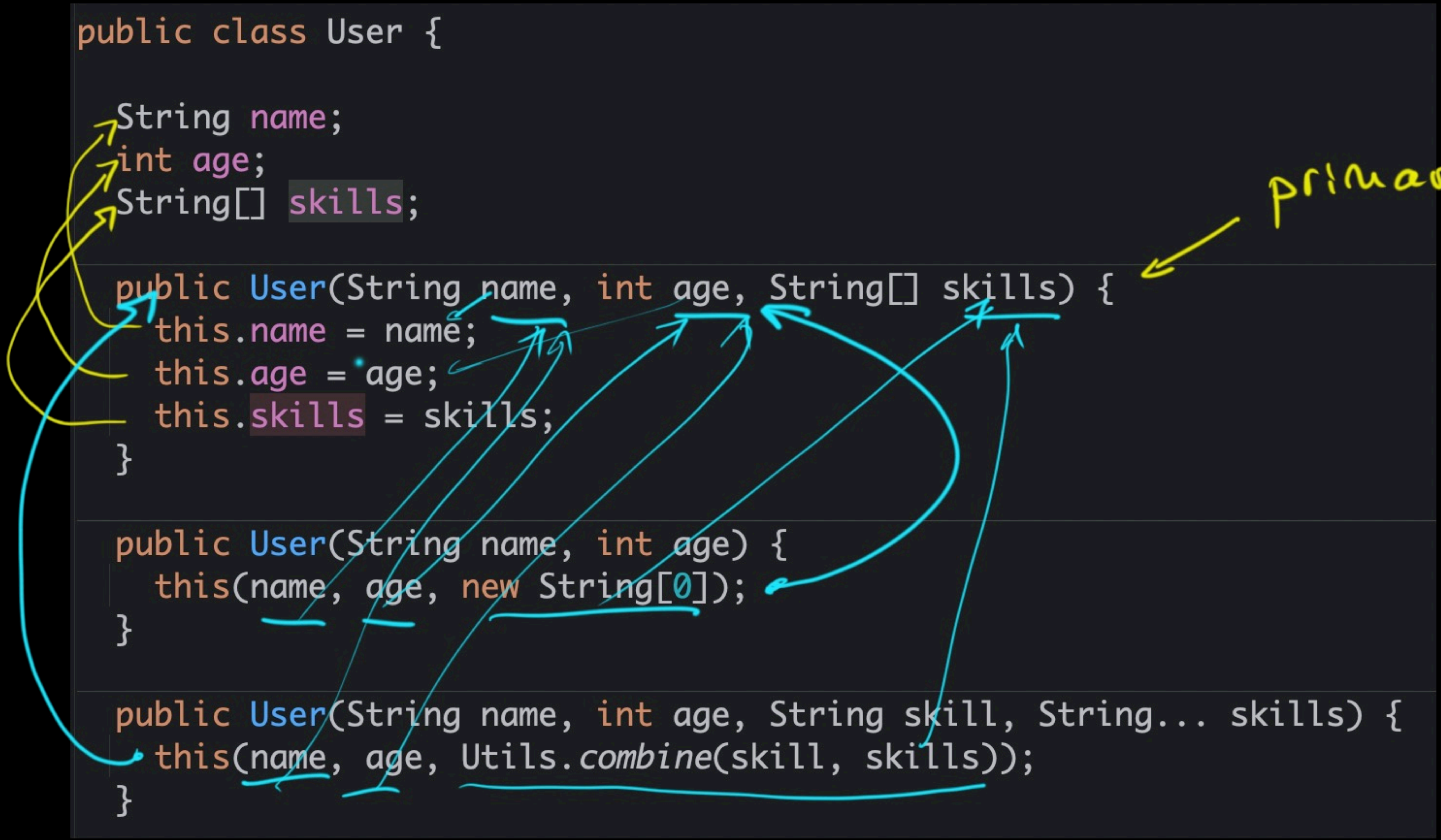
$$\underrightarrow{\text{add (int, int)} \rightarrow \text{Int}}$$



```java
Pizza pizza = new Pizza( name: "Margarita",  size: 30);
```

```java
public class Pizza {

  String name;
  int size;

  //public Pizza() {}
  // constructor
  public Pizza(String name, int size) {
    this.name = name;
    this.size = size;
}
  }
```

```java
String.format("Pizza[name: %s, size: %d]", name, size);
```



```java
public class User {

  String name;
  int age;
  String[] skills;

  public User(String name, int age, String[] skills) {
    this.name = name;
    this.age = age;
    this.skills = skills;
  }

  public User(String name, int age) {
    this(name, age, new String[0]);
  }

  public User(String name, int age, String skill, String... skills) {
    this(name, age, Utils.combine(skill, skills));
  }
```

primary constructor (1)

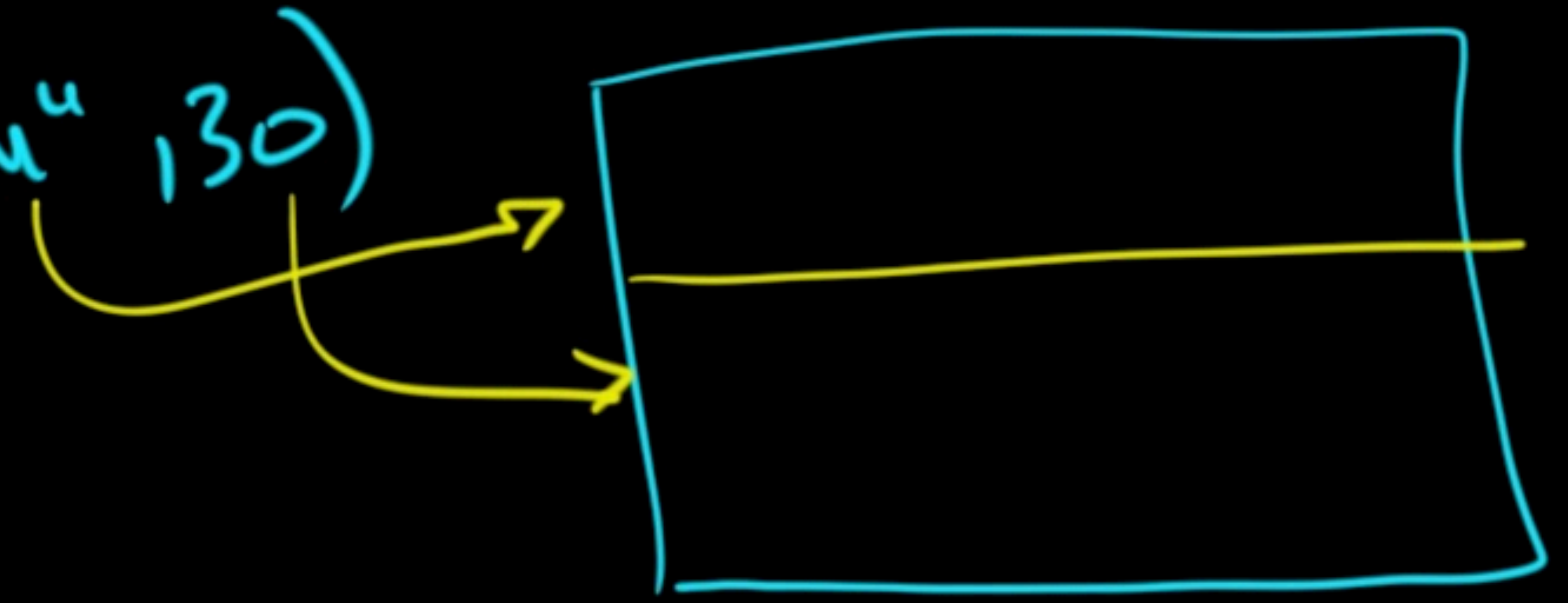
secondary constr

$a + b$

$a \quad -a$

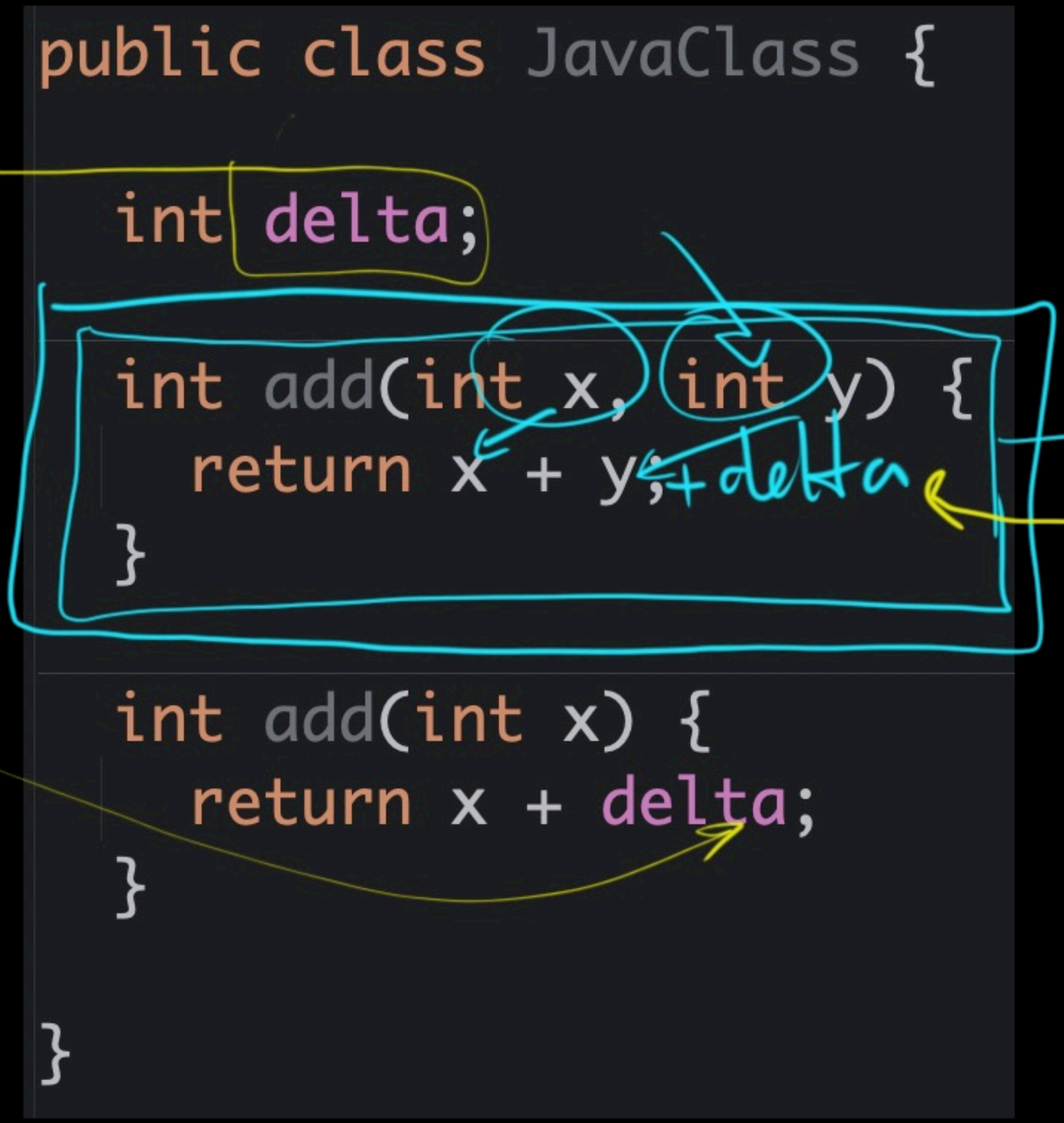
$a == b$

new Pizza("M" 130)



## function vs method

method = f + closure (all class)

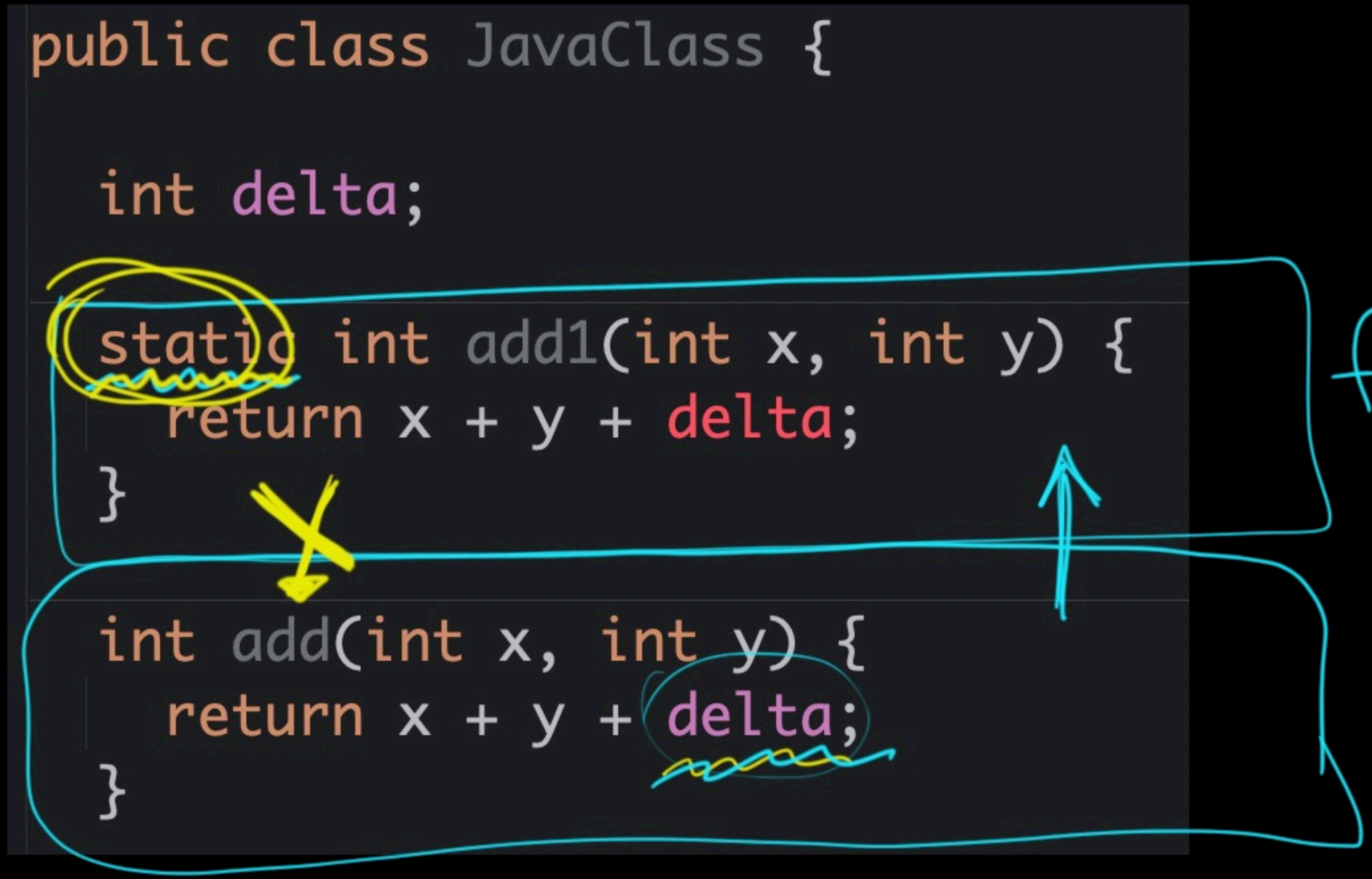


```
public class JavaClass {

  int delta;

  int add(int x, int y) {
    return x + y;+delta
  }

  int add(int x) {
    return x + delta;
  }

}
```

→ function (method)

→ method = function + access to class fields

closure



```java
public class JavaClass {

  int delta;

  static int add1(int x, int y) {
    return x + y + delta;
  }

  int add(int x, int y) {
    return x + y + delta;
  }
```

function = static method = method w/o closure

only params

method = has access to closure

all fields from the class

$$\begin{tikzcd}
\text{byte} \arrow[r] \arrow[dr] \arrow[ddr] & \text{short} \\
& \text{int} \\
& \text{long}
\end{tikzcd}$$

$$\begin{tikzcd}
\text{short} \arrow[r] \arrow[dr] & \text{int} \\
& \text{long}
\end{tikzcd}$$

$\text{int} \rightarrow \text{long}$

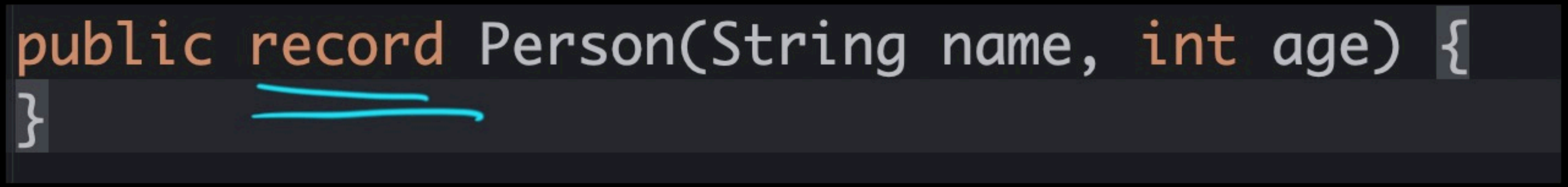
$\text{float} \rightarrow \text{double}$

Pizza extends Object

User extends Object

$\forall \longrightarrow \text{Object}$

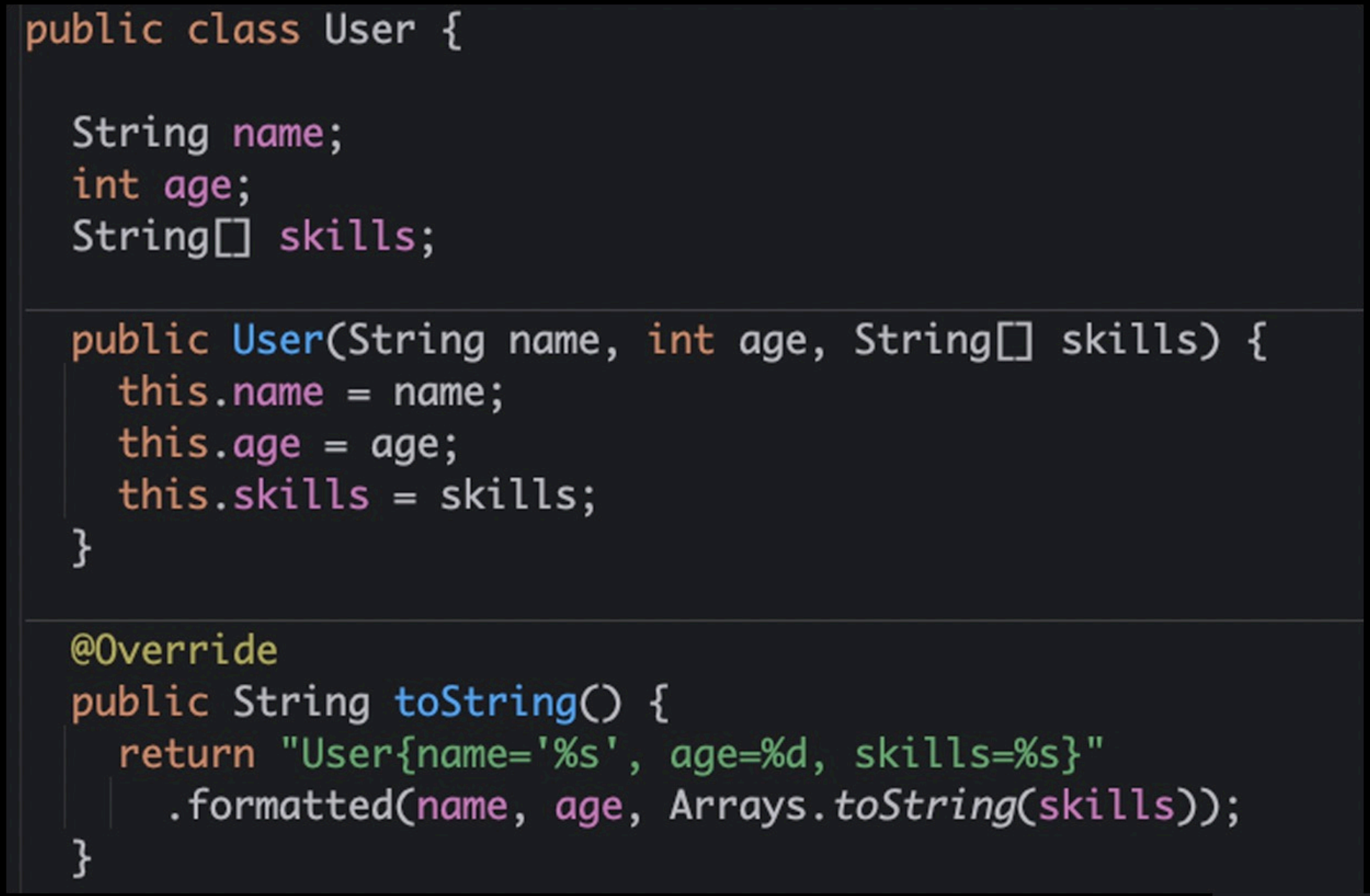
$[\,] \longrightarrow \text{Object}$



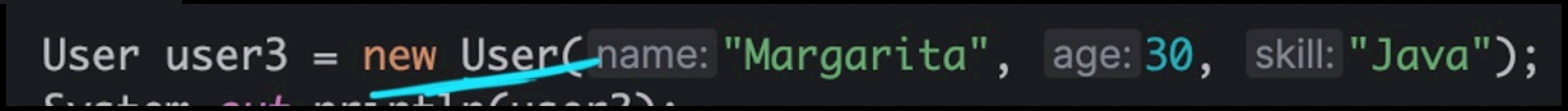
```java
public record Person(String name, int age) {
}
```

17

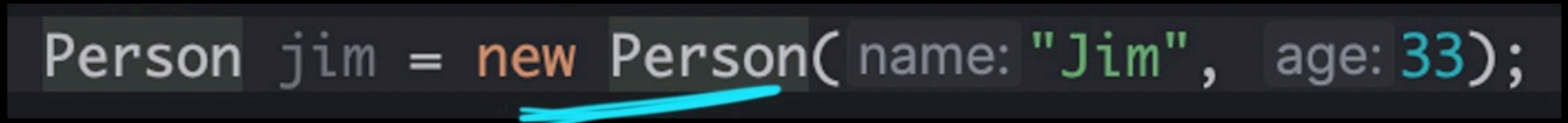
$$\begin{array}{r} 23 \\ -17 \\ \hline 6 \end{array}$$



```java
public class User {

  String name;
  int age;
  String[] skills;

  public User(String name, int age, String[] skills) {
    this.name = name;
    this.age = age;
    this.skills = skills;
  }

  @Override
  public String toString() {
    return "User{name='%s', age=%d, skills=%s}"
      .formatted(name, age, Arrays.toString(skills));
  }
```



```java
User user3 = new User( name: "Margarita",  age: 30,  skill: "Java");
```



```java
Person jim = new Person( name: "Jim",  age: 33);
```