Inline functions

compilation

f

u

n

m

a

i

n

(

)

{

V

a

ι

r

e

S

[redacted]

[redacted]

a

p

p

ζ

y

T

W

C

e

(

2

)

{

i

t

2

}

}

f

u

n

a

p

p

ʅ

y

T

W

i

C

e

(

i

:

I

n

t

,

f

:

(

I

n

t

)

[redacted]

$>$

I

n

t

)

I

n

t

[redacted]

[redacted]

f

(

f

(

)

```java
public static final Integer applyTwice(
        Integer i,
        Function1<Integer, Integer> f) {
    return f.invoke(f.invoke(i));
}
public static final void main() {
    Integer res = applyTwice(2,
        new Function1<Integer, Integer>() {
            @Override
            public Integer invoke(Integer i) {
                return i * 2;
            }
        });
}
```

* pseudo-code that describes the worst case scenario, the compiler might do some extra optimizations where possible

```
fun applyTwice(i: Int, f: (Int) -> Int): Int = f(f(i))
```

```
fun main() {
    val res = applyTwice(2) {
        it * 2
    }
}
```

Inline functions

compilation

```
fun applyTwice(i: Int, f: (Int) -> Int): Int = f(f(i))
```

```
fun main() {
    val res = applyTwice(2) {
        it * 2
    }
}
```

→ *

```
public static final Integer applyTwice(
        Integer i,
        Function1<Integer, Integer> f) {
    return f.invoke(f.invoke(i));
}
public static final void main() {
    Integer res = applyTwice(2,
        new Function1<Integer, Integer>() {
            @Override
            public Integer invoke(Integer i) {
                return i * 2;
            }
        });
}
```

* pseudo-code that describes the worst case scenario, the compiler might do some extra optimizations where possible

Inline functions

compilation

```
fun applyTwice(i: Int, f: (Int) -> Int): Int = f(f(i))
```

```
fun main() {
    val res = applyTwice(2) {
        it * 2
    }
}
```

→ *

* pseudo-code that describes the worst case scenario, the compiler might do some extra optimizations where possible