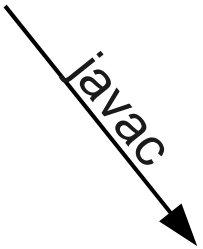
```
void m() {
    int y = 3;
    Function<Integer,Integer> f = x -> x + y;
    f.apply(2);
}
```

Naïve desugaring

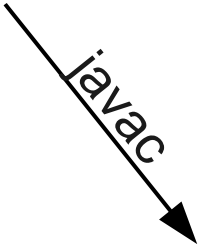
```
void m() {
    int y = 3;
    Function<Integer,Integer> f = x -> x + y;
    f.apply(2);
}
```




```
void m() {
    int y = 3;
    Function<Integer,Integer> f = new A$1(y);
    f.apply(2);
}

class A$1 implements Function<Integer,Integer> {
    private final int y;
    A$1(int y) { this.y = y; }

    public Integer apply(Integer x) {
        return x + y;
    }
}
```

Project Lambda ABI

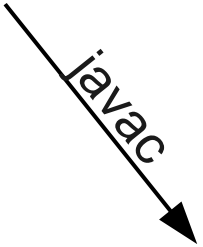
```
void m() {
    int y = 3;
    Function<Integer,Integer> f = x -> x + y;
    f.apply(2);
}
```




```
static Integer lambda$1(int y, Integer x) {
    return x + y;
}

void m() {
    int y = 3;
    Function<Integer,Integer> f = λ-factory;
    f.apply(2);
}
```

Project Lambda ABI

```
void m() {
    int y = 3;
    Function<Integer,Integer> f = x -> x + y;
    f.apply(2);
}
```




```
static Integer lambda$1(int y, Integer x) {
    return x + y;
}

void m() {
    int y = 3;
    Function<Integer,Integer> f =
            INDY[ j.l.i.LambdaMetaFactory,
                  MT[Function.apply],
                  MH[lambda$1]
                ](y);
    f.apply(2);
}
```

INDY mechanics

```
Function<Integer,Integer> f =
    INDY[ j.l.i.LambdaMetaFactory,
          MT[Function.apply],
          MH[lambda$1]
        ](y);
```

INDY mechanics

```
Function<Integer,Integer> f =
    INDY[ j.l.i.LambdaMetaFactory,
          MT[Function.apply],
          MH[lambda$1]
        ](y);
```

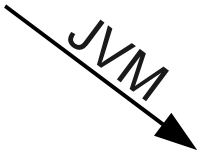

```
1) (once) execute
     j.l.i.LambdaMetaFactory(MT[Function.apply],MH[lambda$1]);

2) (once) store result to
     vmstatic CallSite CS;

3) execute
     Function<Integer,Integer> f = CS.get().invoke(y);
```

HotSpot implementation

```
j.l.i.LambdaMetaFactory(MT[Function.apply],MH[lambda$1])
```

HotSpot implementation

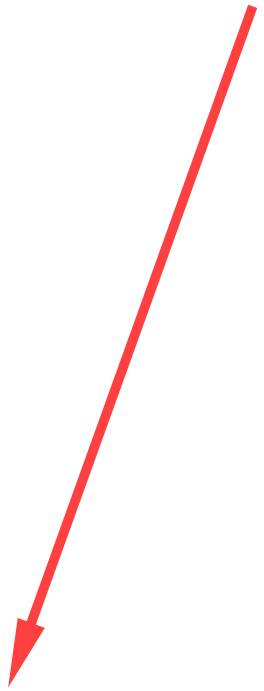
```
j.l.i.LambdaMetaFactory(MT[Function.apply],MH[lambda$1]) {
    generate(...);
```




```
class A$1 implements Function<Integer,Integer> {
    private final int y;
    A$1(int y) { this.y = y; }

    public Integer apply(Integer x) {
        return lambda$1(y, x);
    }
}
```

HotSpot implementation

```
j.l.i.LambdaMetaFactory(MT[Function.apply],MH[lambda$1]) {
    generate(...);
    return new CallSite(MH[A$1#new]);
}
```

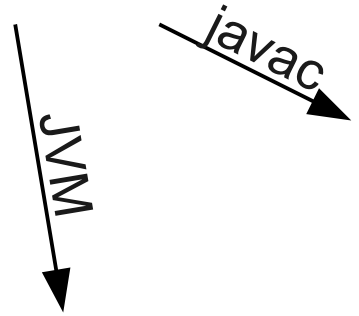


```
class A$1 implements Function<Integer,Integer> {
    private final int y;
    A$1(int y) { this.y = y; }

    public Integer apply(Integer x) {
        return lambda$1(y, x);
    }
}
```

```
CS.get().invoke(y);  ~  return new A$1(y);
```

Non-capturing lambda

```
void m2() {
    Function<Integer,Integer> f = x -> x + 3;
    f.apply(2);
}
```




```
static Integer lambda$23(Integer x) {
    return x + 3;
}
```

```
static A$23$INSTANCE = new A$23();

class A$23 implements Function<Integer,Integer>{

    A$23() {}

    public Integer apply(Integer x) {
        return lambda$23(x);
    }
}
```

```
CS.get().invoke();  ~  return A$23$INSTANCE;
```