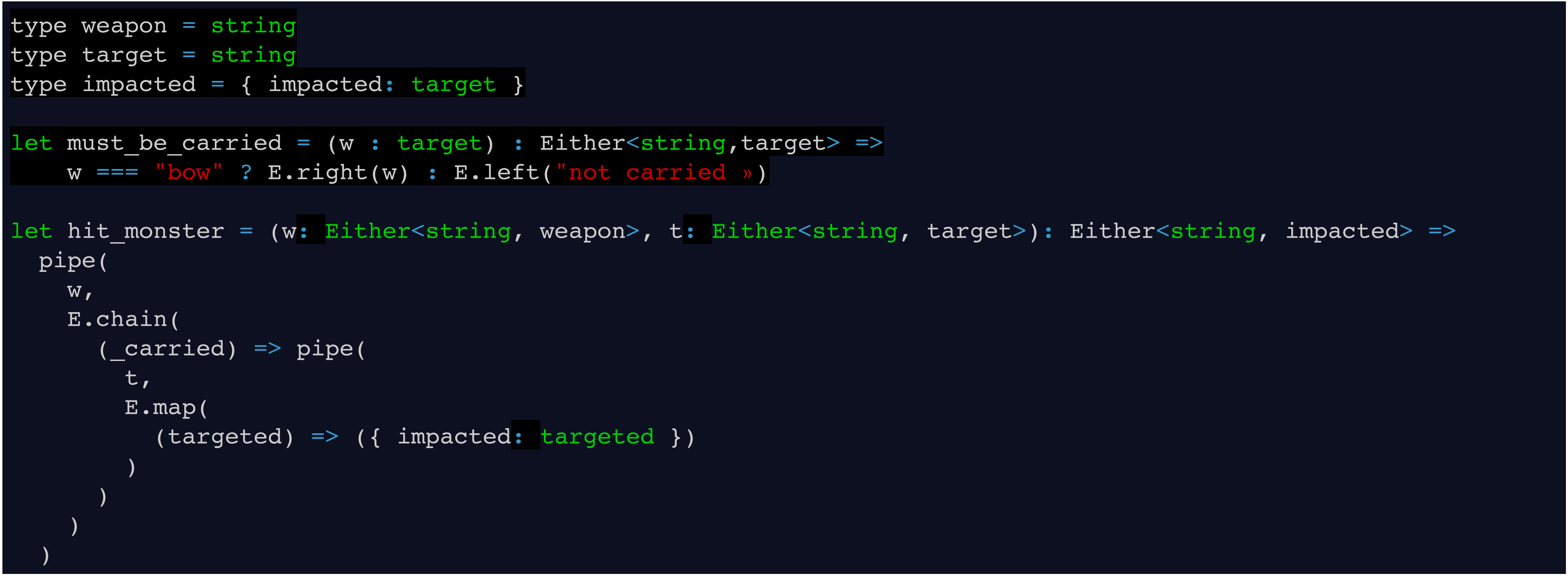
MODÉLISER UNE ERREUR POTENTIELLE

CHAIN (AKA FLATMAP AKA BIND) / MAP

```
chain: <E, A, B>(f: (a: A) => E.Either<E,B>) => (ma: E.Either<E,A>) => E.Either<E,B>

map: <A, B>(f: (a: A) => B) => (fa: E.Either<E,A>) => E.Either<E,B>
```



```
type weapon = string
type target = string
type impacted = { impacted: target }

let must_be_carried = (w : target) : Either<string,target> =>
    w === "bow" ? E.right(w) : E.left("not carried »)

let hit_monster = (w: Either<string, weapon>, t: Either<string, target>): Either<string, impacted> =>
  pipe(
    w,
    E.chain(
      (_carried) => pipe(
        t,
        E.map(
          (targeted) => ({ impacted: targeted })
        )
      )
    )
  )
```

MODÉLISER UNE ERREUR POTENTIELLE

CHAIN (AKA FLATMAP AKA BIND) / MAP

```
type weapon = string
type target = string
type impacted = { impacted: target }

let must_be_carried = (w : target) : Either<string,target> =>
    w === "bow" ? E.right(w) : E.left("not carried »)

let hit_monster = (w: Either<string, weapon>, t: Either<string, target>): Either<string, impacted> =>
  pipe(
    w,
    E.chain(
      (_carried) => pipe(
        t,
        E.map(
          (targeted) => ({ impacted: targeted })
        )
      )
    )
  )
```

```
chain: <E, A, B>(f: (a: A) => E.Either<E,B>) => (ma: E.Either<E,A>) => E.Either<E,B>

map: <A, B>(f: (a: A) => B) => (fa: E.Either<E,A>) => E.Either<E,B>
```

MODÉLISER UNE ERREUR POTENTIELLE

EN JAVA >= 17