```rust
trait Calculate {
    type Error: From<i32>;

    fn calculate(&self, b: i32) -> Result<i32, Self::Error> {
        let a = self.a();

        if a >= b {
            Ok(a - b)
        } else {
            Err(From::from(a + b))
        }
    }

    fn a(&self) -> i32;
}
```

```rust
#[derive(Debug)]
pub struct MyError(i32);

impl From<i32> for MyError {
    fn from(number: i32) -> MyError {
        MyError(number)
    }
}

struct Foo(i32);

impl Calculate for Foo {
    type Error = MyError;

    fn a(&self) -> i32 {
        self.0
    }
}

fn main() {
    let foo = Foo(10);

    println!("{:?}", foo.calculate(3));
    println!("{:?}", foo.calculate(13));
}
```

```
Ok(7)
Err(MyError(23))
```