에러 처리

2020.3

Panic!

- 러스트는 아직 세련된 에러처리에 어려움이 있다.
- Panic!
- 릴리즈 모드에서 스택 되감기 없이 종료

```
[profile.release]
panic = ‘abort'
```

Backtrace 하기
RUST_BACKTRACE=1 cargo run

디버그 심볼은 디버그 모드에서 포함되어 컴파일 됨

Result

- 반드시 성공이 보장되지 않은 연산은 Result로 처리.

```
enum Result<T, E>{
    Ok(T),
    Err(E),
}

use std::fs::File;

fn main() {
    let f = File::open(“hello.txt"); // 파일이 존재하지 않을수도, 억세스 권한이 없을 수도,.....
}
```

에러의 처리

```rust
use std::fs::File;
use std::io::ErrorKind;

fn main() {
    let f = File::open("hello.txt");

    let f = match f {
        Ok(file) => file,
        Err(ref error) if error.kind() == ErrorKind::NotFound => {
            match File::create("hello.txt") {
                Ok(fc) => fc,
                Err(e) => {
                    panic!(
                        "Tried to create file but there was a problem: {:?}",
                        e
                    )
                },
            }
        },
        Err(error) => {
            panic!(
                "There was a problem opening the file: {:?}",
                error
            )
        },
    };
}
```

Unwrap, expect

- Ok 내의 값을 반환하거나 Err이면 panic! 호출

```
use std::fs::File;

fn main() {
    let f = File::open("hello.txt").unwrap();
}
```

```
use std::fs::File;

fn main() {
    let f = File::open("hello.txt").expect("Failed to open hello.txt");
}
```

에러의 전파

```
use std::io;
use std::io::Read;
use std::fs::File;

fn read_username_from_file() -> Result<String, io::Error> {
    let f = File::open("hello.txt");

    let mut f = match f {
        Ok(file) => file,
        Err(e) => return Err(e),
    };

    let mut s = String::new();

    match f.read_to_string(&mut s) {
        Ok(_) => Ok(s),
        Err(e) => Err(e),
    }
}
```

? 연산자

```rust
use std::io;
use std::io::Read;
use std::fs::File;

fn read_username_from_file() -> Result<String, io::Error> {
    let mut f = File::open("hello.txt")?;
    let mut s = String::new();
    f.read_to_string(&mut s)?;
    Ok(s)
}

fn read_username_from_file() -> Result<String, io::Error> {
    let mut s = String::new();

    File::open("hello.txt")?.read_to_string(&mut s)?;

    Ok(s)
}
```

안전한 값

```
pub struct Guess {
    value: u32,
}

impl Guess {
    pub fn new(value: u32) -> Guess {
        if value < 1 || value > 100 {
            panic!("Guess value must be between 1 and 100, got {}.", value);
        }

        Guess {
            value
        }
    }

    pub fn value(&self) -> u32 {
        self.value
    }
}
```