Noter til "Error Handling"

## 2. Agenda

- Hvad er værre end at gå ned?
- Fejlhåndtering i din kode
- Fejltolerante arkitekturer
- Krigshistorier (real-life eksempler)

## 3. Hvad er værre end at gå ned?

- Rangering af crash scenarier fra bedst til værst:
    . Applikation virker som forventet og går aldrig ned.
    . Applikation går ned på grund af sjældne fejl, som ingen bemærker eller bekymrer sig om.
    . Applikation går ned på grund af en almindeligt forekommende fejl.
    . Applikation dødvæger og stopper med at svare på grund af en almindelig fejl.
    . Applikation går ned længe efter den oprindelige fejl.
    . Applikation forårsager datatab og/eller korruption.

## 4. Fejlhåndtering i din kode

- Brug kun undtagelser, du kan håndtere.
- Undgå at bruge undtagelser til at håndtere logiske fejl som `NullReferenceException`.
- Brug ikke undtagelser til at emulere kontrolflow.

## 5. Compiler hjælp med fejlhåndtering

- Brug af `out` parametre i C#:

```
csharp
Kopier kode
public static bool TryParse(string s, out int result);
```

- Oprettelse af typer, der indfanger fejl:

```
csharp
Kopier kode
public abstract class Option<T> {
    public abstract bool Exists();
    public abstract T Get();
}

public class Some<T> : Option<T> {
    private T t;
    public Some(T t) { this.t = t; }
    public override bool Exists() => true;
    public override T Get() => t;
}

public class None<T> : Option<T> {
    public override bool Exists() => false;
    public override T Get() => throw new Exception();
}
```

**6. Dynamiske typelanguages**

- Eksempler: JavaScript, PHP, Python
- Variabler kan tildeles og omfordeles til noget som helst.
- Ingen kontrol for returtype af funktioner matcher.

**7. Linting og Advarsler**

- Advarsler kommer fra compileren – lyt til dem!
- Linting er et separat værktøj (eks. Resharper, Sonarlint).
- Behandl altid advarsler som fejl – de er advarsler af en grund.

**8. Fejltolerante Arkitekturer**

- Håndtering af fejl gennem:
    . Redundans (tænk Netflix).
    . Reduceret kapabilitet.
- Test af fejlhåndtering: Chaos Engineering.

**9. Krigshistorier**

- Forvent skrald, når du interfacer med andre systemer eller brugerinput, og håndter det ved at sanitisere dit input.
- Strategier ved hukommelsesudløb:
    . Crash.
    . Fang fejlen og alarmer brugeren (hvis muligt).
    . Fang fejlen og prøv at fortsætte.
    . Aldrig løb tør for hukommelse.