Gruppemedlemmer:

Emil Berglund

obaida

Aram

Herman Jacobsen

Oppgave 1:

1.1:

Exceptions handler om å håndtere feil og forhindre at programmet krasjer.

Vær spesifikk i exception-handling ved å håndtere de situasjonen som kan skje. Eksempelvis bruk ArgumentNullException, ikke ArgumentException hvis man forventer null-error.

1.2:

1. Vi har opplevd problemer i andre fag, men ikke i Rammeverk og .NET. Alle feilmeldinger vi har fått, har vært lette å forstå.

Konsekvensene av dårlige exceptions er forvirring for utviklere, og dårlig brukeropplevelse for bruker, ved at programmet kan krasje. Dårlige exceptions leder til mye feilsøkingstid for utviklere, som kunne vært unngått ved bedre feilhåndtering.

2 og 3.

* Feilhåndtering
* En sjekk som sier hva som har gått feil
* Unngå uklarhet
* Unngå forvirring
* Unngå programkrasj

1.3:

En exception håndterer feil eller uventede situasjoner under kjøring av et program. Ved å implementere exceptions kan man unngå forvirring, at progammet krasjer og behandle disse feilene deretter.

Oppgave 2:

2.1.

2.2.

Scenario A: *ArgumentOutOfRangeException* eller *IndexOutOfRangeException*

Scenario B: *ArgumentOutOfRangeException, NullReferenceException, KeyNotFoundException, ArgumentNullEx, ObjectNotFoundException, ArgumentException*

Scenario C: *HttpRequestException, TimeoutException, System.AccessViolationException, InvalidOperationException*

Scenario D: *FormatException, IndexOutOfRangeException*

Bonus: Lag en custom-exception

Begrunnelse:

Vi har valgt ut følgende exceptions, ettersom det er disse vi har sett for hos fungerer i ulike situasjoner. Vi har også sett for oss at custom-exceptions vil være lurt her, ettersom det er noen tilfeller som kan håndteres bedre med egne feilhåndteringer og sjekker.

Oppgave 3:

3.1 –

3.2.

Et bilde som inneholder tekst, skjermbilde, programvare, datamaskin

KI-generert innhold kan være feil.

1,2,3: -

4.

Vi navnga først klassen InvalidCreditCardException. Dette endret vi til InvalidTransactionException, for et mer universelt navn. Klassen har vi basert på Oppgave 2, Scenario D hvor det handlet om feil format med at nummeret måtte være tallet 4 eller 5.

Oppgave 4:

4.1 –

4.2.1 Unødvendig kompleksitet, lager ting som allerede fins, det som finnes er basert på godt kodegrunnlag. Dersom vi lager egne exceptions kan vi utelukke små faktorer som har stor betydning for koden, som den eksisterende exceptions pakken allerede har inkludert. Koden blir også lettere å forstå for andre utviklere, siden de eksisterende pakkene er velkjent og utviklere slipper å sette seg inn i ny kode.

4.2.2. Feilmeldinger må være presise, konkrete, men ikke oppgi sensitiv informasjon. En god feilmelding kan se slik ut*: Customer ID -5 is invalid. ID must be a positive integer.*

Feilmeldingen skal påpeke hva som er feil, og fortelle hvordan man unngår feilen.

4.2.3 Exceptions bør brukes når metoder ikke kan fullføre sin oppgave på grunn av interne eller eksterne feil, for eksempel at en fil ikke finnes. For forventede scenarier, som at en bruker angir feil passord, er det ofte bedre å returnere en statuskode eller en boolsk verdi.

4.2.4 For flytkontroll er det bedre å bruke logikk fremfor exceptions ettersom det krever ekstra CPU og minnehåndtering, og man kan noen ganger oppleve vanskeligere debugging, som leder til at kontrollflyten kan bli uklar.

4.2.5 Det er viktig å dokumentere hvilke exceptions en metode kan kaste, slik at utviklere ve hva de kan forvente og hvordan de kan håndtere dem. Dette forbedrer kodekvaliteten og gjør det lettere å vedlikeholde systemet over tid.

4.3

Ja, etter diskusjonene har definisjonen blitt tydeligere. I starten var fokuset kanskje kun på å kaste exceptions, men nå ser vi at god exception-håndtering handler om når og hvordan de brukes, samt hvordan man kan gjøre koden mer robust og lesbar.

Nye innsikter:

* Exceptions bør kun brukes på uventede feil, ikke for flytkontroll
* Gjenbruk av standard exceptions gjør koden mer forståelig og enklere å vedlikeholde.
* Klare og presise feilmeldinger hjelper både utviklere og brukere.

Best practices:

* Bruk eksisterende exceptions, fremfor å lage custom hvor det ikke er nøvendig
* Gode og tydelige navn og feilmeldinger
* Velg flere exceptions som er forventet, ikke én som tar alt.
* Ikke vis sensitiv informasjon i feilmeldinger

Kilder:

<https://powershellexplained.com/2017-04-07-all-dotnet-exception-list/>

<https://learn.microsoft.com/en-us/dotnet/standard/exceptions/best-practices-for-exceptions>

<https://stackoverflow.com/questions/28970912/what-exception-should-be-thrown-for-an-invalid-id>