POLYMORFI I OBJEKTORIENTERET PROGRAMMERING

OBEJEKTORIENTERET PROGRAMMERING: POLYMORFI

Polymorfi er et begreb i programmering, der dækker over flere ting, men idéen er at bruge samme grænseflade til forskellige datatyper – det kan give koden noget fleksibilitet!

POLYMORFI = MANGE FORMER
Det at man kan tilgå forskellige 'ting' i programmet på den samme måde, med netop samme metode-kald.
Men hvordan opnå denne polymorfi og fleksibilitet?

VIA

OVERRIDE af baseklassens metode

METHOD OVERLOADING, flere versioner af samme metode (nogen mener, at det ikke er polymorfi, men et godt koncept!)

OBEJEKTORIENTERET PROGRAMMERING: POLYMORFI

!

OVERRIDE AF METODER: dynamisk/runtime polymorfi

Eksempel:
Vi har en Array-Liste af ‘Shapes’, og

OBEJEKTORIENTERET PROGRAMMERING: POLYMORFI

OVERRIDE AF METODER: dynamisk/runtime polymorfi

1: Se projektet 'polymorfiButtton' – de forskellige **ButtonKlasser is a ButtonBase**
optegn klassediagram og udpeg også, hvordan hver enkelt subklasse overrider baseklassens metoder, og gør videre rede for hvordan dette bruges på arrayListen.
Prøv også at ændre på metode-navnet overButton() i en subklasse:
Hvilken fejlmelding får man allerede, inden programmet kører?

2: Se munter video med mr. Coding Train om samme emne og med lignende eksempel:
https://www.youtube.com/watch?v=qqYOYIVrso0

3: Lav lille projekt med arvehierarki, som på en eller anden måde anvender polymorfi

OVERLOAD AF METODER: statisk/compiler polymorfi

(Men nogen mener som sagt, at det ikke er en del af polymorfi)

Overloading giver mulighed for at have forskellige versioner af samme metode i samme klasse. Altså samme metodenavn, men med forskellig signatur og indhold!

```
void udskrivTekst(String s) {
        //…
}

void udskrivTekst(String s, int posX, int posY) {
        //…
}
```

Samme metodenavn, men forskellig signatur og forskellig adfærd!
Parametrene bestemmer, hvilken version der kaldes…

OVERLOAD AF METODER: statisk/compiler polymorfi

(Men nogen mener som sagt, at det ikke er en del af polymorfi)

Samme metodenavn, men forskellig signatur og forskellig adfærd!
Parametrene bestemmer, hvilken version der kaldes…

1: Åben projektet 'overloadingEksempel'

Se de to metoder med samme navn og tilføj en tredje metode –
bare lige for at prøve det selv…

OBEJEKTORIENTERET PROGRAMMERING: POLYMORFI

OVERRIDE AF METODER: dynamisk/runtime polymorfi

- Override af metoder fra Object!
- Alle objekter, lavet ud fra 'class', er i virkeligheden subklasser af Object!
- Og med Object kommer der en række basismetoder, som fx toString()

1: Se projektet 'overrideObject' –
Udpeg hvordan toString-metoden bliver overridet.
Se hvilke øvrige metoder, der på forhånd ligger i klassen, selvom den ikke umiddelbart er subklasse af en baseklasse!

PS: Man skriver navnet på objekt + punktum, og så foreslår Processing de metoder, som hører til objektet…

OBEJEKTORIENTERET PROGRAMMERING: POLYMORFI – fleksibilitet i koden

Override af metode fra baseklassen
Overload af metode i samme klasse

Implementering af metode fra interface
Klassen kan så instantieres som interface-typen, og dermed kan man kalde samme metode på en instans uafhængigt af hvilken klasse det egentlig er, så længe det er bundet op på det specifikke interface som en slags kontrakt om at skulle implementere en eller flere metoder!

Downcasting & upcasting
Kalde baseklassens overridede metode inden i den metode, der overrider via keywordet, super.metodekald()

Projekttype:
Kast terning – ryst()

CodingTrain, partikelsystem