Static vs Dynamic binding
Override vs Overload

Objekt-orienterad programmering och design

Alex Gerdes och Sólrún Halla Einarsdóttir, 2018

Quiz: Gissa typen?

```
Object  o = new Square(100,100);
Polygon p = new Square(200,200);
Square  s = new Square(300,300);
```

Vilken typ har `o`, `p` respektive `s`?

… Är du helt säker?

Static type vs Dynamic type

- En referens-variabel har en *declared type* – detta är dess *statiska* typ, och den kommer alltid att vara densamma (därav ordet statisk).

- En referens-variabel refererar till ett objekt i heapen. Det objektets typ är variabelns *dynamiska* typ. Denna kan förändras (därav ordet dynamisk), om variabeln ges ett nytt referensvärde som pekar på ett annat objekt med en annan typ.

Quiz: Svar

```
Object  o = new Square(100,100);
Polygon p = new Square(200,200);
Square  s = new Square(300,300);
```

`o`, `p` och `s` har statiska typer `Object`, `Polygon` respektive `Square`.

Alla tre har (för närvarande) dynamisk typ `Square`.

Compile time vs run time

- Typinformation används vid två olika tillfällen:
  - Kompilatorn utför *statisk typkontroll (static type checking*). Denna information används enbart vid kompileringen, för att garantera att allt kommer att gå rätt när programmet körs, och för att välja rätt metod(signatur). Informationen om statisk typ sparas inte efter kompileringen.
  - Under exekvering görs *dynamisk typkontroll (dynamic type checking),* och den dynamiska typen används för att bestämma vilka metoder som faktiskt körs.

Overloading

- Overloading innebär att ett objekt (eller class) har flera metoder med samma namn, men olika signatur (dvs typer på dess parametrar).

```
String.subString(int beginIndex);
String.subString(int beginIndex, int endIndex);
```

- Quiz: Hur och när bestäms vilken av signaturerna som används?
- Svar: Det bestäms *statiskt*, vid kompilering.

Overriding

- Overriding innebär att en subklass kan definiera en metod med samma namn och signatur som en metod i dess superklass.

```
Polygon.paint(Graphics g);
Triangle.paint(Graphics g);
```

- Quiz: Hur och när bestäms vilken av metoderna som används?
- Svar: Det bestäms *dynamiskt*, vid exekvering.

Override eller Overload?

- Antag att vi har följande:

```
Polygon.scale(int factor);
Rectangle.scale(int factorX, int factorY);
```

- Quiz: Är detta overriding eller overloading?
- Svar: Overloading: Eftersom `Rectangle` inte definierar en egen `scale(int factor)` ärvs denna från `Polygon`, och `Rectangle` har således två `scale`-metoder med olika signatur.

Diagnostiskt prov

- På nästa föreläsning: jag kommer att fråga er om ett antal metodanrop, liknande övningen, och vill att ni ska svara rätt på vilken metod som används i varje fall.

- Provet är fullständigt harmlöst och bara för er egen skull (jag kommer inte att samla in resultat).

Övning

- Utgå från resultatet från föregående övning (kan laddas ner färdig från hemsidan om ni inte har er egen).
- Implementera `toString()`-metoder för Polygon och alla dess subclasses. Det viktiga är att de skriver ut vilken typ objektet i fråga har.
- Lägg till en metod `overlaps(Polygon p): boolean` i `Polygon`. Vi låtsas att denna metod beräknar huruvida två polygoner överlappar. För nu, låt den alltid returnera `true` – men skriva ut (till `System.out`) tillräcklig information för att kunna avgöra vilken metod som anropas, och vilken typ operanderna har:
  - E.g. ”Polygon.overlaps: Square vs Triangle” – använd `toString()`.
- Skriv en `main`-metod som skapar några olika polygoner av olika slag och jämför med varandra med `overlaps`. Stämmer utskrifterna med vad ni förväntar dig? Kan ni förklara vad som händer i termer av statisk och dynamisk typ?
- Lägg till en overloaded metod `overlaps(Triangle t): boolean` i `Triangle`, och motsvarande i övriga subclasses. Låt dessa skriva ut liknande information så ni kan se vilken metod som anropas. Testa igen att jämföra olika polygoner. Stämmer det med vad ni förväntar er?
- Skapa ytterligare en polygon (av någon specifik dynamisk typ), men låt denna hållas av en variabel av statisk typ `Polygon`. Testa att jämföra även denna, med sig själv och med övriga polygoner med mer specifik statisk typ. Testa den både som det objekt som `overlap` anropas på, och som argument. Kan ni förutsäga vilka utskrifter ni kommer få?
- Overridea metoden `overlaps(Polygon p): boolean` i alla subclasses, med en särskiljande utskrift. Kör testerna igen. Kan ni förutsäga vilka utskrifter ni kommer få?