PGR112 –Oppgaver 08 (2023-02-01)

Før du leser videre etter denne setningen, pakk ut .ZIP-filen og utforske klassene og forholdene mellom klassene.

Dette oppgavesettet går ut på å lage ulike figurer som er geometriske: Sirkler, kvadrater, rektangler, osv.

Målet er at vi i et senere oppgavesett vil tegne disse grafisk ut i et vindu hvor vi vil se disse figurene visuelt, hvor dette oppgavesettet går ut på å klargjøre disse klassene for bruk senere, det vil si at objektene som opprettes som instanser har dataene og metodene disse figurene trenger for å tegnes senere.

I kodeutgangspunktet for oppgavene, finner vi 3 klasser:

- Shape
- Circle
- Square

Begynn med å lese igjennom disse klassene før du fortsetter.

I .java filene hentet ut fra .ZIP-filen, arver både Circle og Square fra klassen Shape. I forelesningen har vi hatt om abstrakte klasser, og en del av oppgavene går ut på å ta i bruk nøkkelordet **abstract** og tilpasser koden i de ulike klassene deretter.

Akkurat nå er det mulig å opprette instanser (objekter) av klassen Shape, noe som kan virke lite hensiktsmessig, fordi vi har jo klasser som representerer faktiske figurer.

Ved å opprette Shape-objekter, så vil ikke disse objektene kunne representere spesifikke geometriske figurer, da klassen i seg selv er laget for å representere figurer på en generisk måte.

- Oppgave #1
    - Sørg for at vi ikke kan opprette objekter av type Shape

- Oppgave #2
    - Vi ønsker oss at alle ulike subklasser av klassen Shape (klasser som arver fra Shape) tilbyr følgende metoder:
        - En metode som regner ut og returnerer arealet til figuren
        - En metode som regner ut og returnerer omkretsen til figuren

      Hint: Metoder kan være abstrakte!

      Gjør nødvendige endringere for å få dette til

- Oppgave #3
    - Opprett en Rectangle-klasse, som har følgende felt for å holder på data om det geometriske tilknyttet figuren:
        - double width
        - double height

        Sørg for at denne klassen implementerer metodene fra oppgave #2!

- Oppgave #4
    - Gjør om Square-klassen slik at denne arver fra Rectangle-klassen istedenfor Shape-klassen.

        Fjern overflødige felter og metoder, da disse vil kunne arves direkte fra Rectangle-klassen istedenfor! Sjekk gjerne om BlueJ/IntelliJ rapporterer om dette i form av feilmeldinger, og ta det deretter.

- Oppgave #5
    - Sørg for at alle figurer har et **id**-felt av datatypen **int** som ikke kan endres etter den er satt.

        Lag en getter-metode for å hente ut denne identifikator verdien.

        Hint: et statisk felt i en klasse som alle figurene forholder seg til kan brukes

- Oppgave #6
    - I alle sub-klassene, opprett en toString() metode for å kontrollere hvordan informasjon om objektet printes ut til terminalen dersom det forsøkes å printe ut referansen til objektet direkte til terminalen.

      Legg til @Override annotasjonen på disse metodene

- Oppgave #7
    - I en main-metode, opprett 10 objekter av forskjellige geometriske figurer, og plasser disse i et HashMap hvor id-verdien fungerer som nøkkel.

- Oppgave #8
    - Gå igjennom alle objektene (figurene) i HashMap'en og skriv ut informasjon om hvert objekt til terminalen.

- Oppgave #9
    - Velg ut et objekt fra HashMap'en basert på en nøkkel du vet eksisterer. Sjekk om at du fikk forventet objekt ved å skrive ut informasjon om objektet.

Ekstra utfordringer:

a) Det finnes flere ulike måter å gå igjennom alle elementene i et HashMap på. Ta en titt på linken du finner nederst.

   Forsøk de ulike metodene mtp. oppgave #8 og sjekk ut hvilken måte du foretrekker å bruke.

b) Skriv ut informasjon om alle figurer som har et areal som er større enn en viss grense.

   Lagre denne grensen i en variabel og bruk denne i løsningen din.

Informasjon om ulike måter å bruke HashMap:

- https://www.w3schools.com/java/java_hashmap.asp