Almuthanna Jamal Aljajan

Teori

Oppgave 1.1

Lag deg en oversikt over hva følgende ord/begreper betyr, *med egne ord* (lurt å gjøre dette bra nå, kan være nyttig til eksamen):

- **Class:**

  En kasse Kan sees på som en "blueprint" oppskrift for et element i programmet vårt.
  Generalisert er klasse bygd opp av:Variabler , konstuktur og Metoder eksempel på hvordan man definerer en klasse.

  ```
  public class Planet {

  private String navn;

   private double radius;

  private double gravitasjon;}
  ```

- **Object (konseptet, ikke klassen);**

  Et objekt er en realisering / instanse av klasse, eksempel på hvordan man oppretter et objekt.

  ```
  Planet marsPlanet = new Planet();

  marsPlanet.setNavn("Mars");

  marsPlanet.setRadius(3389.5);

  marsPlanet.setGravitasjon(3.711);
  ```

- **Instansvariabel:**

  Instansvariabeler deklareres i en klasse, men utenfor en metode, konstruktør eller en hvilken som helst blokk. De opprettes når et objekt opprettes ved bruk av nøkkelordet 'new'. Instansvariabeler er synlige for alle metoder, konstruktører og blokker i klassen. Normalt anbefales det å gjøre disse variablene private. De har standardverdier. For tall er standardverdien 0, for Booleans er den falsk, og for objektreferanser er den null. Verdier kan tilordnes under erklæringen eller i konstruktøren.

  ```
  class Page {

  public String pageName;
  ```

```
// instance variable with public access
private int pageNumber;
// instance variable with private access
}
```

- **Overloading:**

  Overloading av metoder gjør at det er mulig for en klasse å ha mer enn én metode som har samme navn, hvis argumentlistene er forskjellige. Det ligner konstruktør-overloading i Java, som gjør at en klasse kan ha mer enn en konstruktør som har forskjellige argumentlister.

- **Overriding:**

  Overriding gjør at det er mulig for en sub-klasse for å gi en bestemt implementering av en metode som allerede er gitt ved en av sine superklasser. Når en metode i en underklasse har samme navn, samme parametere eller signatur og samme returtype som en metode i superklassen, sies metoden i sub-klassen overrider metoden i super-klasse.

- **Extends**

  Nøkkelordet extends indikerer at en klasse er arvet fra en annen klasse. I Java er det mulig å arve attributter og metoder fra en klasse til en annen.

  Sub-klasse (barn) - klassen som arver fra en annen klasse

  Superklasse (foreldre) - klassen som blir arvet fra

- **Polymorphism**

  betyr "mange former", og det oppstår når vi har mange klasser som er knyttet til hverandre etter arv.

- **private,public,(protected) (klasse,variabel,metode)**

  Private Access Modifier Hvis en metode eller variabel er merket som privat, er det bare kode i samme klasse som har tilgang til variabelen, eller kalle metoden. Kode i sub-klasser har ikke tilgang til variabelen eller metoden, og kan heller ikke kode fra noen ekstern klasse.

  Public access Modifier betyr at all kode kan få tilgang til klassen, feltet, konstruktøren eller metoden, uavhengig av hvor tilgangskoden befinner seg. Tilgangskoden kan være i en annen klasse og annen pakke.

  Protocted access Modifier

Variabler, metoder og konstruktører, som er deklarert protocted i en superklasse, kan bare nås av sub-klassene i annen pakke eller hvilken som helst klasse i pakken til de protocted medlemmers klassen.

- **this og super**

  this og super: når en metode blir kalt, blir den automatisk sendt et implisitt argument som er en henvisning til det på kaller objektet, denne referansen kalles this.

  Super: vi kan bruke super på flere måter.

  En Superclass kan kalle en konstruktør definert av sin superclass ved bruk av formen super, for eksempel (super (parameterliste)); Her, parameterliste spesifiserer alle parametere som trengs av konstruktøren i Superclass.

  Det er en annen form for super som bruker super for å få tilgang til superklasse medlemmer, bortsett fra at den alltid refererer til superclass til subclass den brukes i. Denne bruken har følgende generelle form: (super.member), her kan medlem være enten en metode eller forekomstvariabel. Denne formen for superklasse er mest anvendelig for situasjon der medlemsnavn på subclass skjuler medlemmer med samme navn i superclass