Klasser og Objekter

Kursus 3

Objekt Orienteret Programmering (OOP)

- UML
  - Strandard til at moduler klasser, objekter og adfærd
  - Skabelon fra use case til implementation

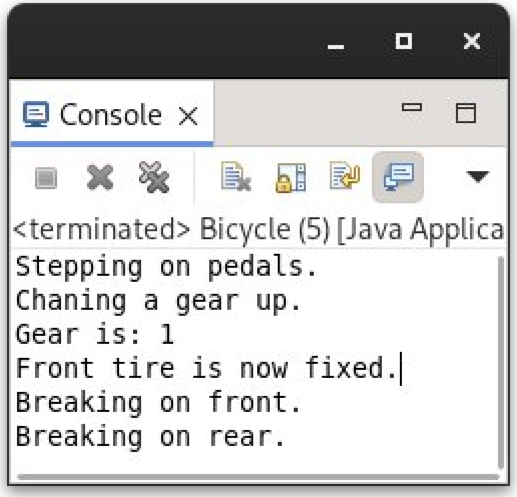
- Klasser
  - Skabelon til at definere adfærd
  - Metoder vs. attributter

- Objekt
  - En realisering af en klasse
  - Pointer fra stack til heap hukommelse

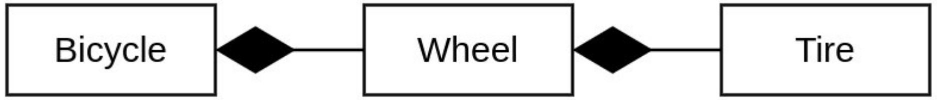
Opgave 4: Run Bicycle

- Main method
  - En del af *Bicycle*
  - Synlighed: *Public vs Pivate*

- UML:
  - Composition:
    - En cykel kan ikke virke uden hjul og dæk



```
Stepping on pedals.
Chaning a gear up.
Gear is: 1
Front tire is now fixed.
Breaking on front.
Breaking on rear.
```



```java
public static void main(String args[]) {
    // Create tires for both front and rear wheels
    Tire frontTire = new Tire();
    Tire rearTire = new Tire();
    Wheel frontWheel = new Wheel(frontTire);
    Wheel rearWheel = new Wheel(rearTire);

    // Create the Bicycle with wheels
    Bicycle myBicycle = new Bicycle(frontWheel, rearWheel);

    // "I can ride it forward by stepping on the pedals"
    myBicycle.peddel();

    // "by changing gears I can go faster"
    myBicycle.changeGearUp();

    // "I start in first gear."
    System.out.println("Gear is: " + myBicycle.gear); // 1

    // "have front and backlights which switch on if it becomes dark"
    Boolean dark = true;
    if (dark) {
        myBicycle.frontLight = true;
        myBicycle.rearLight = true;
    }

    // "The wheels I can take off when I need to fix holes in the tire."
    myBicycle.frontWheel.tire.flat = true;
    if (myBicycle.frontWheel.tire.flat) {
        myBicycle.frontWheel.tire.repair();
        System.out.println("Front tire is now fixed.");
    }

    // "When I brake, I can use both front and read brakes"
    Boolean breaking = true;
    while (breaking) {
        myBicycle.pressBreakFront();
        myBicycle.pressBreakRear();
        breaking = false;
    }
}
```