Primitives, Object References, Static, and Non-Static Variables

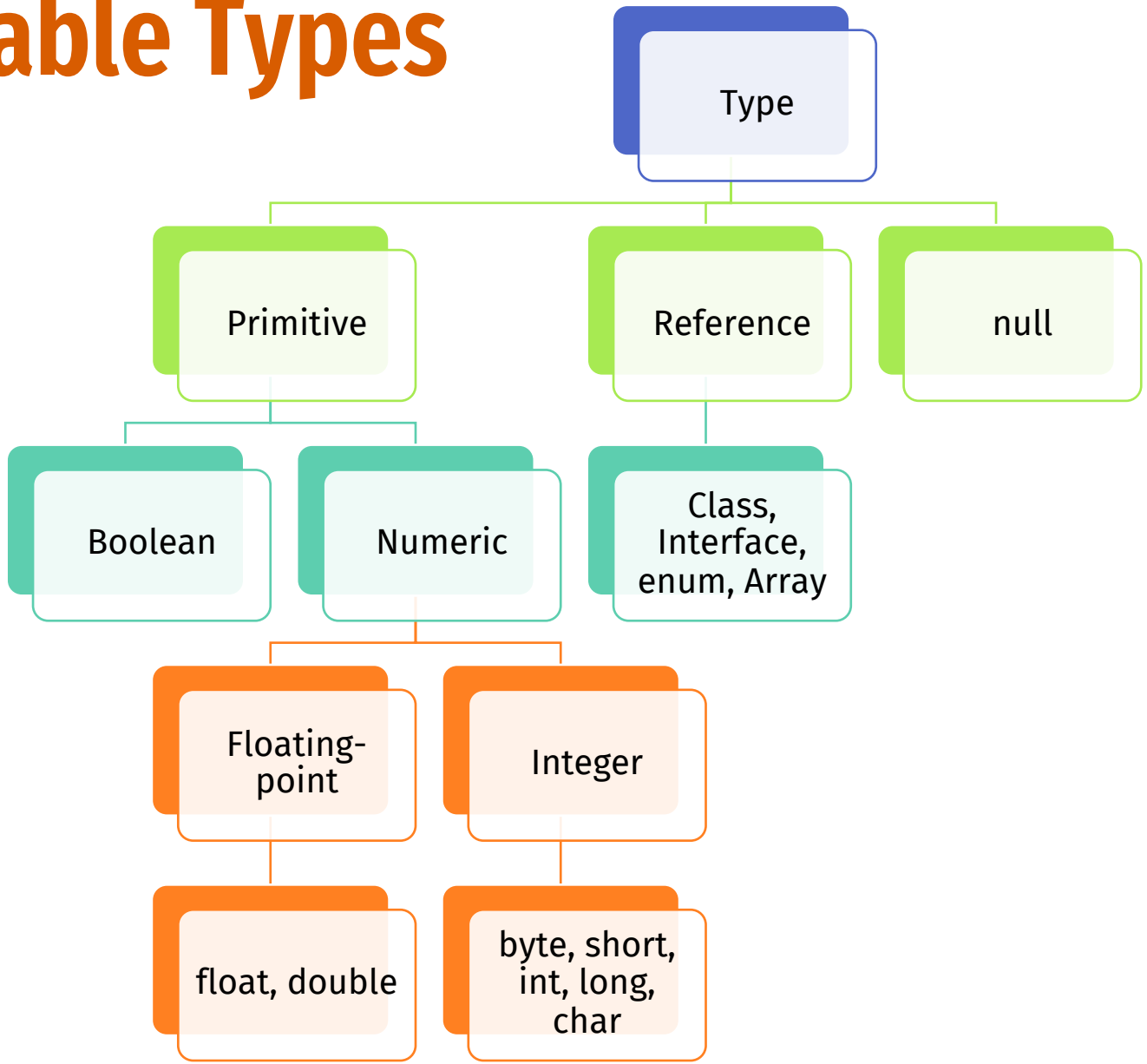
Object-oriented Programming

Aryo Pinandito, S.T., M.MT, Ph.D.

Variable Types (Java)

- Primitive
- Reference
- null value

Variable Types



Default Value of Variable Type

Type	Default Value
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
string (or any Object)	Null
boolean	false

Example

```
class Character {
  int hp; // 0
  float atk; // 0.0f
  boolean isSleeping; // false
}
```

Example

```
class Character {
  int hp;
  float atk;
  boolean isSleep;
}
```

	Memory
hp	0
atk	0.0
isSleep	false

Example

```
class Character {
  int hp = 450;
  float atk = 52.6;
  boolean isSleep = true;
}
```

Primitive variables always have value.
Primitive variables cannot have null value.

	Memory
hp	450
atk	52.6
isSleep	true

Primitive vs Reference Variable

- The value of primitive variables are stored in stack memory
- The value of reference variables are stored in heap memory
  - The references itself are kept in stack memory
  - Reference variable have a default null value, which means no data on heap memory is referenced (pointed).

Reference Variable Value

```
class Character {
  int hp;
  Weapon wpn;
  Item itm;
}
```

```
Character c =
  new Character();
```

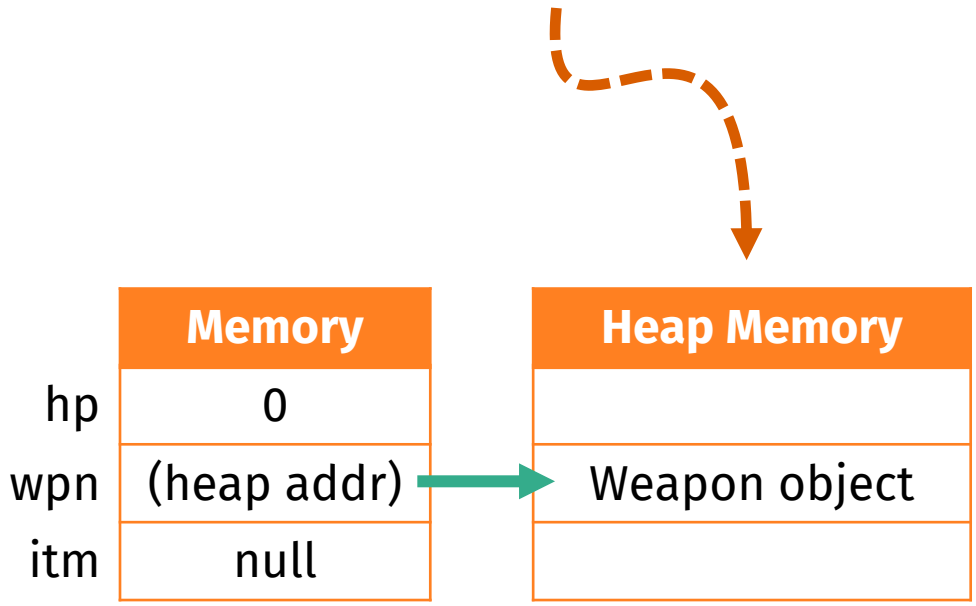
	Memory
hp	0
wpn	null
itm	null

Heap Memory

Reference Variable Value

```
class Character {
  int hp;
  Weapon wpn;
  Item itm;
}
```

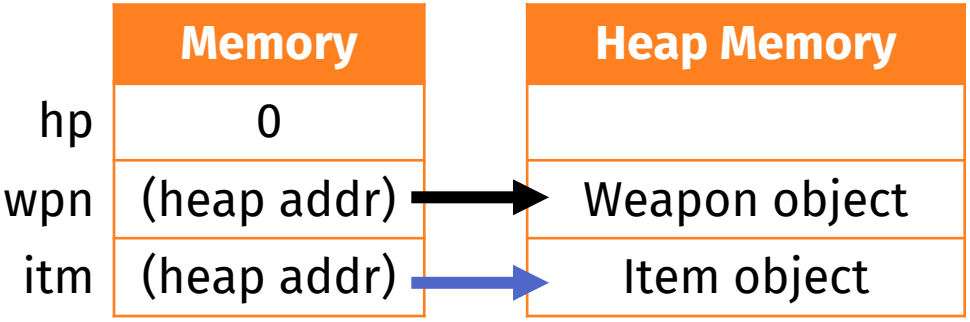
```
Character c =
  new Character();
c.wpn = new Weapon();
```



Reference Variable Value

```
class Character {
  int hp;
  Weapon wpn;
  Item itm;
}
```

```
Character c =
  new Character();
c.wpn = new Weapon();
c.itm = new Item();
```



Passing Value

Primitive Variable

- Assigning a value to a primitive data type, the value is **copied**.
- The same thing occurred when passing a primitive value when calling a method.
- Change in the value of one primitive variable does not affect another primitive variable's value.

Example

```
int x;
```

	Memory
x	0

Example

```
int x = 11;
```

	Memory
x	11

Example

```
int x = 11;
int y;
```

	Memory
x	11
y	0

Example

```
int x = 11;
int y = x;
```

copy!

	Memory
x	11
y	11

Example

```
int x = 11;
int y = x;

x = 8;
```

	Memory
x	8
y	11

Example

```
int x = 11;
int y = x;

x = 8;
y = 42;
```

	Memory
x	8
y	42

Example

```
int plus2(int a) {
  int b = a + 2;
  return b;
}

int x = 3;
```

	Memory
a	0
b	0
x	3

Example

```
int plus2(int a) {
  int b = a + 2;
  return b;
}

int x = 3;
x = plus2(x);
```

copy!

	Memory
a	3
b	0
x	3

Example

```
int plus2(int a) {
  int b = a + 2;
  return b;
}

int x = 3;
x = plus2(x);
```

	Memory
a	3
b	5
x	3

Example

```
int plus2(int a) {
  int b = a + 2;
  return b;
}

int x = 3;
x = plus2(x);
```

copy!

	Memory
a	3
b	5
x	5

Reference Variable

- Assigning a value to a reference data type, only the reference value is **copied**.
  - The reference stores a memory reference to the actual data on heap memory.
- The actual data, which a reference variable is referring (pointing) to, is **not copied**.
- When two or more reference variables refer to the same reference object, changes to the actual data of a reference will be reflected to another reference.

Example

```
class Character {
  int hp;
  Weapon wpnA;
  Weapon wpnB;
}
```

```
Character c = new Character();
```

	Memory
hp	0
wpnA	null
wpnB	null

Heap Memory

Example

```
class Character {
  int hp;
  Weapon wpnA;
  Weapon wpnB;
}
```

```
Character c = new Character();
c.wpnA = new Weapon("Sword");
```

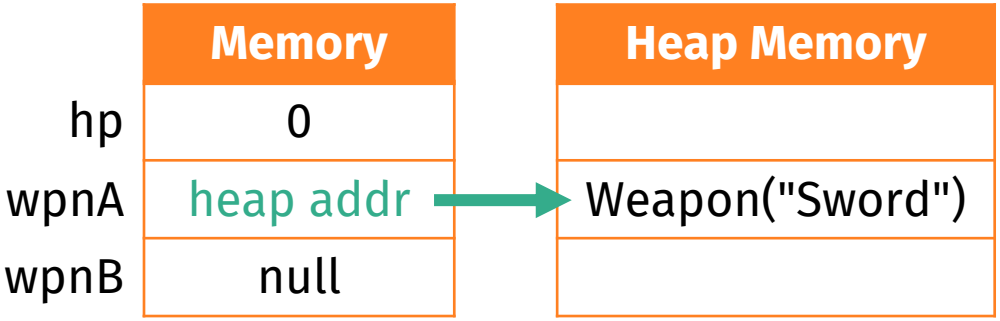
	Memory
hp	0
wpnA	null
wpnB	null

Heap Memory
Weapon("Sword")

Example

```
class Character {
  int hp;
  Weapon wpnA;
  Weapon wpnB;
}
```

```
Character c = new Character();
c.wpnA = new Weapon("Sword");
```



Example

```
class Character {
  int hp;
  Weapon wpnA;
  Weapon wpnB;
}
```

```
Character c = new Character();
c.wpnA = new Weapon("Sword");
c.wpnB = c.wpnA;
```

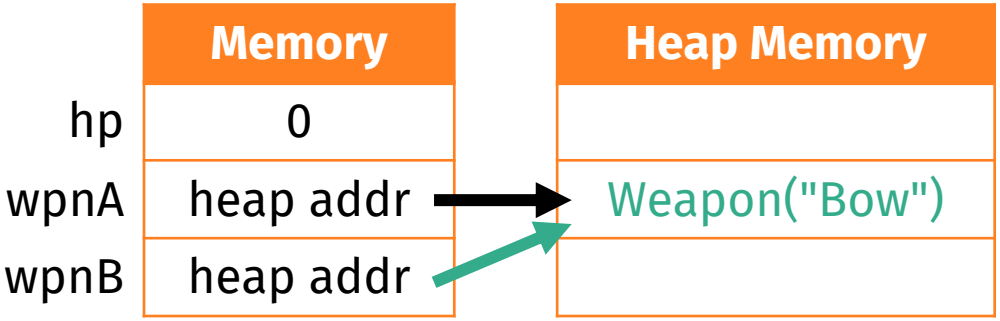
copy the
reference value

	Memory	Heap Memory
hp	0	
wpnA	heap addr →	Weapon("Sword")
wpnB	heap addr →	

Example

```
class Character {
  int hp;
  Weapon wpnA;
  Weapon wpnB;
}
```

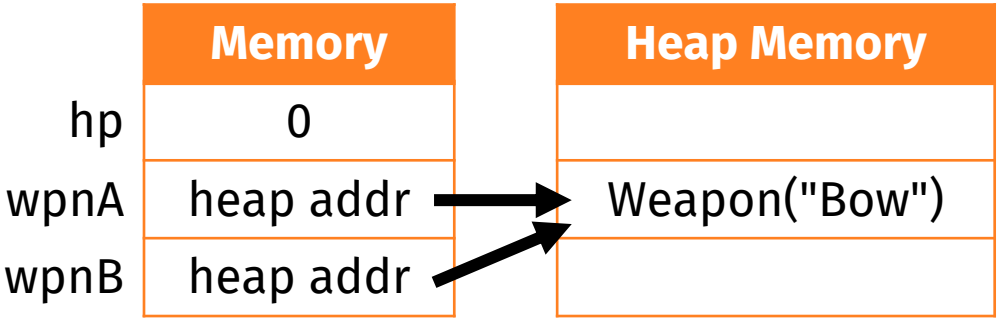
```
Character c = new Character();
c.wpnA = new Weapon("Sword");
c.wpnB = c.wpnA;
c.wpnB.upgrade("Bow");
```



Example

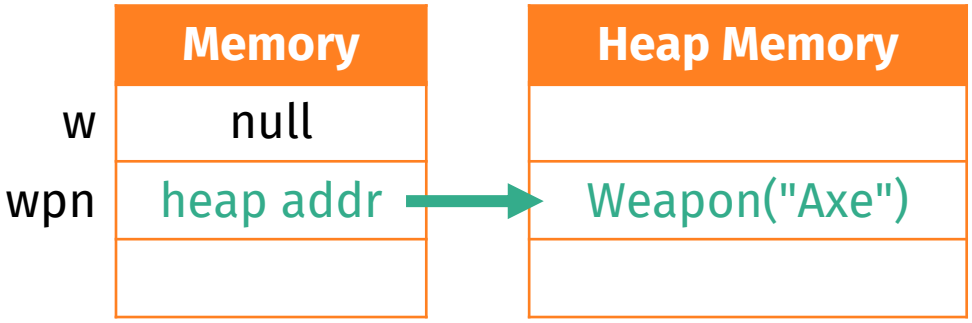
```
class Character {
  int hp;
  Weapon wpnA;
  Weapon wpnB;
}
```

```
Character c = new Character();
c.wpnA = new Weapon("Sword");
c.wpnB = c.wpnA;
c.wpnB.upgrade("Bow");
c.wpnA.print(); // Bow
```



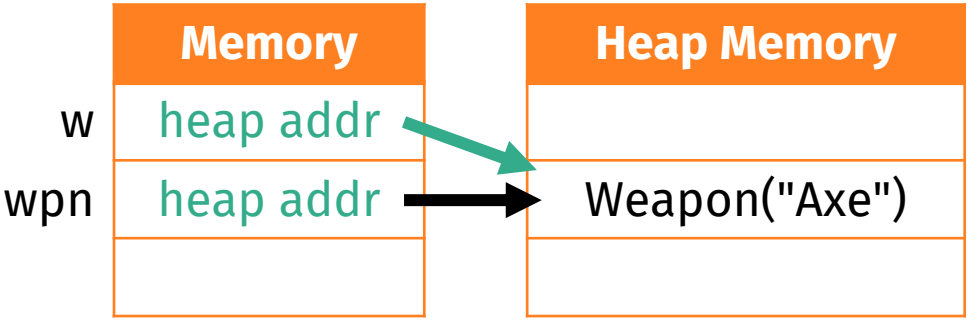
Example

```
void upgrade(Weapon w) {
  w.forgeTo("Bow");
}

Weapon wpn = new Weapon("Axe");
```



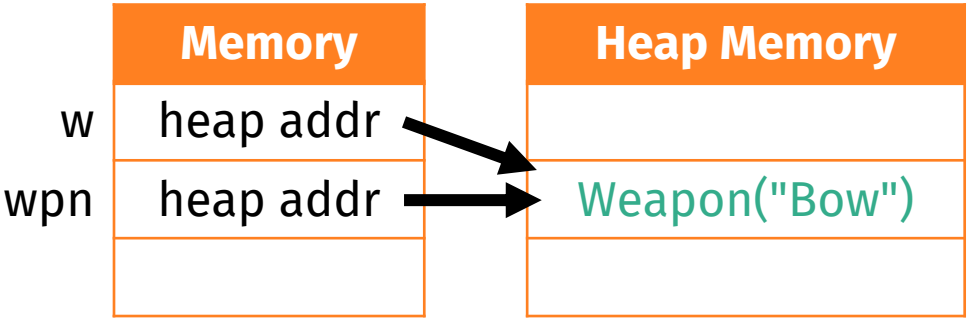
Example

```
void upgrade(Weapon w) {
  w.forgeTo("Bow");
}

Weapon wpn = new Weapon("Axe");
upgrade(wpn);
```



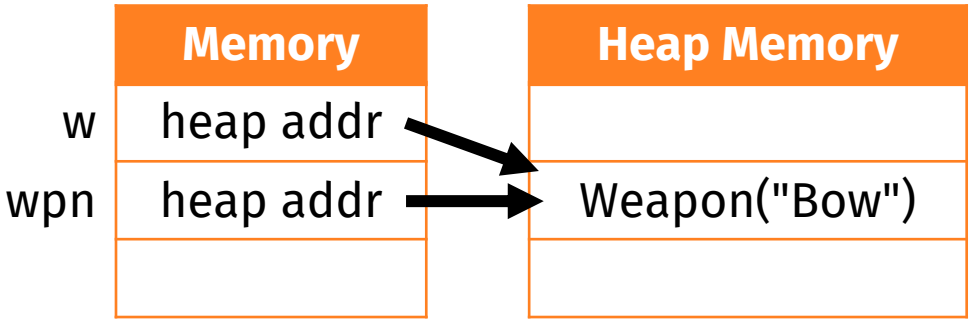
Example

```
void upgrade(Weapon w) {
  w.forgeTo("Bow");
}

Weapon wpn = new Weapon("Axe");
upgrade(wpn);
```



Example

```
void upgrade(Weapon w) {
  w.forgeTo("Bow");
}

Weapon wpn = new Weapon("Axe");
upgrade(wpn);
wpn.print(); // Bow
```



Static and Non-Static Variables

Non-Static Variable and Method

- Non-static (instance) variables and methods belong to instance object.
- Non-static (instance) variables and methods can be accessed or invoked by an instance object of the class.
- Changes to instance variables of an object does not be reflected to another object of the same class.
- An object must be instantiated to access its instance variables or invoke its instance method.

Static Variable and Method

- Static variables are shared by all instances of the class
  - They belong to every object of the class
  - Changes to static variables will be reflected to all objects of the same class
- Static methods are not tied to a specific object
  - Static methods cannot access non-static (instance) variables
- Object instantiation is not necessary to access static variables and static methods

Static Variable and Method

- To declare static variables, constants, and methods, use the **static** modifier.
- Static methods and variables are **underlined** in class diagram.

Weapon
+ <u>baseAtk: int</u>
+ <u>description(): String</u>

```
class Weapon {
  static int baseAtk;
  static String
    description() {
      return "...";
    }
}
```

Accessing Static Variables and Methods

- Static methods and variables can be accessed by both its class or its instance objects.
- Accessing static variables or method from instance object is not recommended.

```
class Weapon {
  static int baseAtk;
  static String
    description() {
      return "...";
    }
}

Weapon.baseAtk = 99;
Weapon.description();

print(Weapon.baseAtk);

Weapon w = new Weapon();
// do not do this!
print(w.baseAtk);
```

Accessing Static Variables and Methods

Static methods cannot access non-static variables

```
class Weapon {

  int atk = 24;
  static int baseAtk;

  static int hit(Character ch) {
    ch.hp = ch.hp - Weapon.baseAtk + atk;
  }

}
```

OK!

Error!

Accessing Static Variables and Methods

Non-static methods can access static variables

```
class Weapon {
  int atk = 24;
  static int baseAtk;
  int hit(Character ch) {
    ch.hp = ch.hp - Weapon.baseAtk + atk;
  }
}
```

OK!

```
Character c = new Character();
Weapon wpn = new Weapon();
wpn.hit(c);
```

Static Constants

Static constants are **final** variables and also shared by all the instances of the class.

```
class Mate {
  public static final double PI = 3.141593;
}

double area = Mate.PI * 7 * 7;
print(area);
```

Questions?

Assignment

Berdasarkan assignment sebelumnya, tambahkan atribut-atribut berikut ini pada class Hero dan Enemy (jika belum ada):

hp: int	Health point dari karakter Hero dan Enemy
def: int	Nilai attack defense dari Hero dan Enemy, default = 0.
baseAtk: int	Base attack point dari Hero dan Enemy, nilai default Hero = 58, Enemy = 46.
weapon: Weapon	Weapon yang digunakan oleh Hero dan Enemy untuk menyerang lawan.
level: int	Level dari Hero dan Enemy, default 1.

Assignment

Pada Weapon, tambahkan atribut dan method berikut ini (jika belum ada):

atk: int	Nilai attack yang diberikan oleh weapon tertentu
name: String	Nama weapon
isBroken: boolean	Status kerusakan weapon.
condition: int	Level kondisi dari weapon (0-100), default 100

repair()	Mengembalikan level kondisi senjata ke 100
use(): int	Menggunakan senjata, mengurangi level kondisi sebesar 10 poin dan mengembalikan nilai atk dari senjata ini
<u>randomAtk (Weapon w): int</u>	Menghasilkan nilai attack random 10-30% dari nilai atk Weapon yang diberikan via parameter method.

Assignment

Pada Hero, tambahkan method berikut ini (jika belum ada):

attack(en: Enemy): void	Merepresentasikan perilaku Hero ketika menyerang musuh. Ketika menyerang, Hero menggunakan senjata yang dipegangnya dan mengurangi nilai hp lawan dengan rumus: level (Hero) * baseAtk (Hero) + atk (Weapon) - def (Enemy) + randomAtk (Weapon); Menghilangkan nilai def dari Enemy.
defense()	Menaikkan nilai poin defense sebesar: baseAtk * level / 2.
heal(): void	Menaikkan hp Hero sebesar 100.

Assignment

Pada Enemy, tambahkan method berikut ini (jika belum ada):

attack(en: Enemy): void	Merepresentasikan perilaku Enemy ketika menyerang musuh. Ketika menyerang, Enemy menggunakan senjata yang dipegangnya dan mengurangi nilai hp lawan dengan rumus: level (Enemy) * baseAtk (Enemy) + atk (Weapon) - def (Hero) + randomAtk (Weapon); Menghilangkan nilai def dari Hero.
defense()	Menaikkan nilai poin defense sebesar: baseAtk * level / 2.
remedy(): void	Menaikkan hp Enemy sebesar 100.

Assignment

- Buat kode program (Java)-nya.
  - Class, atribut, dan method
  - Constructor untuk inisialisasi nilai atribut
  - Instansiasi objek, pemanggilan method
  - Akses terhadap atribut static
  - Pemanggilan method static
- Gambarkan *class diagram*-nya.