BASIC PROGRAMMING WITH JAVA

BASIC SYNTAX

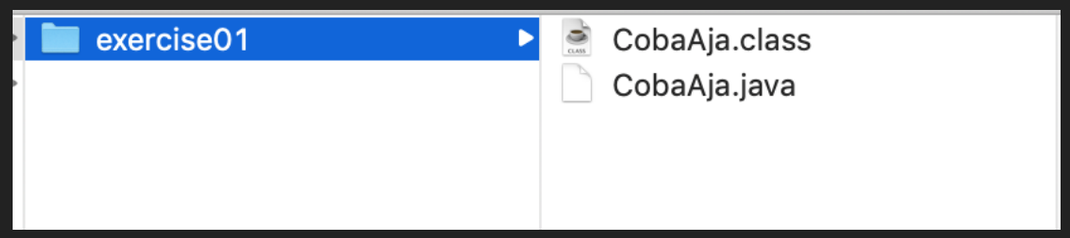
```
public class MyFirstJavaProgram {

  public static void main(String []args) {
    System.out.println("Hello World"); // prints Hello World
  }
}
```

```
class CobaAja{
  public static void main(String[] args){
    String nama = "Angga";
    String nomorTlp="08170998245";
    System.out.println(nama);
    System.out.println("Nama Saya :"+nama);
    System.out.println(nomorTlp);
  }
}
```

- Nama class menggunakan PascalCase
- Apa yg dimaksud dengan ***main*** method
- Method menggunakan camelCase dan verb
- Nama variable menggunakan camelCase

COMPILE & RUN

```
→  exercise01 javac CobaAja.java
→  exercise01
```



CobaAja.java ● CobaAja.class ×

```
1   ???????4?,
2   ????  ??
3   ???
4   ???
5   ??
6   ??? ?!?<init>?()V?Code?LineNumberTable?main?([Ljava/lang/String;)V?
7   SourceFile?CobaAja.java?
8   ??Angga?08170998245?"?#?$?%??&?'?java/lang/StringBuilder?Nama Saya :?(?)?*?+?CobaAja?java
9   ??????????*??????????????    ???????Z??????.LM??+??????Y????? +?? ??
10  ?????,????????????????????
11  ?
12  ??&??-?
13  ??????
```

PROBLEMS TERMINAL ••• 1: zsh

```
→  exercise01 javac CobaAja.java
→  exercise01 java CobaAja
Angga
Nama Saya :Angga
08170998245
→  exercise01
```

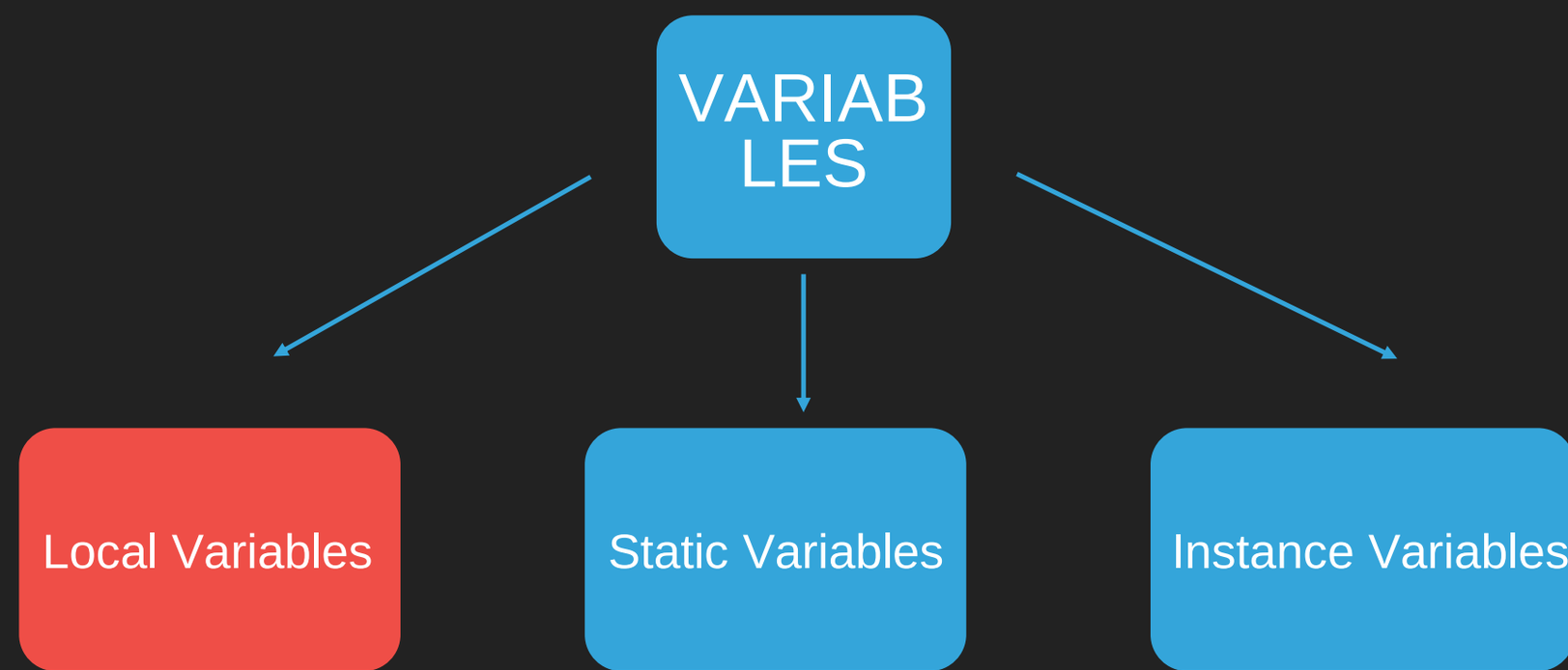
- Javac melakukan proses compile menjadi executable binary
- Executable binary yg dihasilkan terdapat pada file dengan extention .class
- Perintah java untuk menjalankan executable binary / .class nya

JAVA KEYWORDS

Apa jadi nya kalau kita mendeklarasikan variable dengan nama yg sama dengan salah satu keywords

abstract	assert	boolean	break
byte	case	catch	char
class	const	continue	default
do	double	else	enum
extends	final	finally	float
for	goto	if	implements
import	instanceof	int	interface
long	native	new	package
private	protected	public	return
short	static	strictfp	super
switch	synchronized	this	throw
throws	transient	try	void
volatile	while		

VARIABLES



Untuk basic programming kita focus menggunakan local variable

PRIMITIVE DATA TYPE

- **byte** pasti berisi bilangan bulat dengan size -128 sd 127
- **short** bilangan bulat dengan size -32768 sd 32767
- **int** bilangan bulat dengan size -2147483648 sd 2147483647
- **long** bilangan bulat dengan size -2^63 sd 2^62
- **float** single-precision 32-bit IEEE 754 floating point
- **double** double-precision 64-bit IEEE 754 floating point
- **boolean**…… are you kidding me?
- **char** single 16-bit Unicode character

NON-PRIMITIVE DATA TYPES (REFERENCE DATA TYPES)

- Hampir semua tipe data esperti String, Integer (yg berawalan hurry besar) adalah type data Reference
- Mereka memiliki Class Object nya
- Array
- Collections dan turunan nya

OPERATORS

Arithmetic Operators

+ - x / % ++ --

Relational

<, >, ==, <=, >=, != , menhasilkan boolean

Logical Operators

&&, ||, !

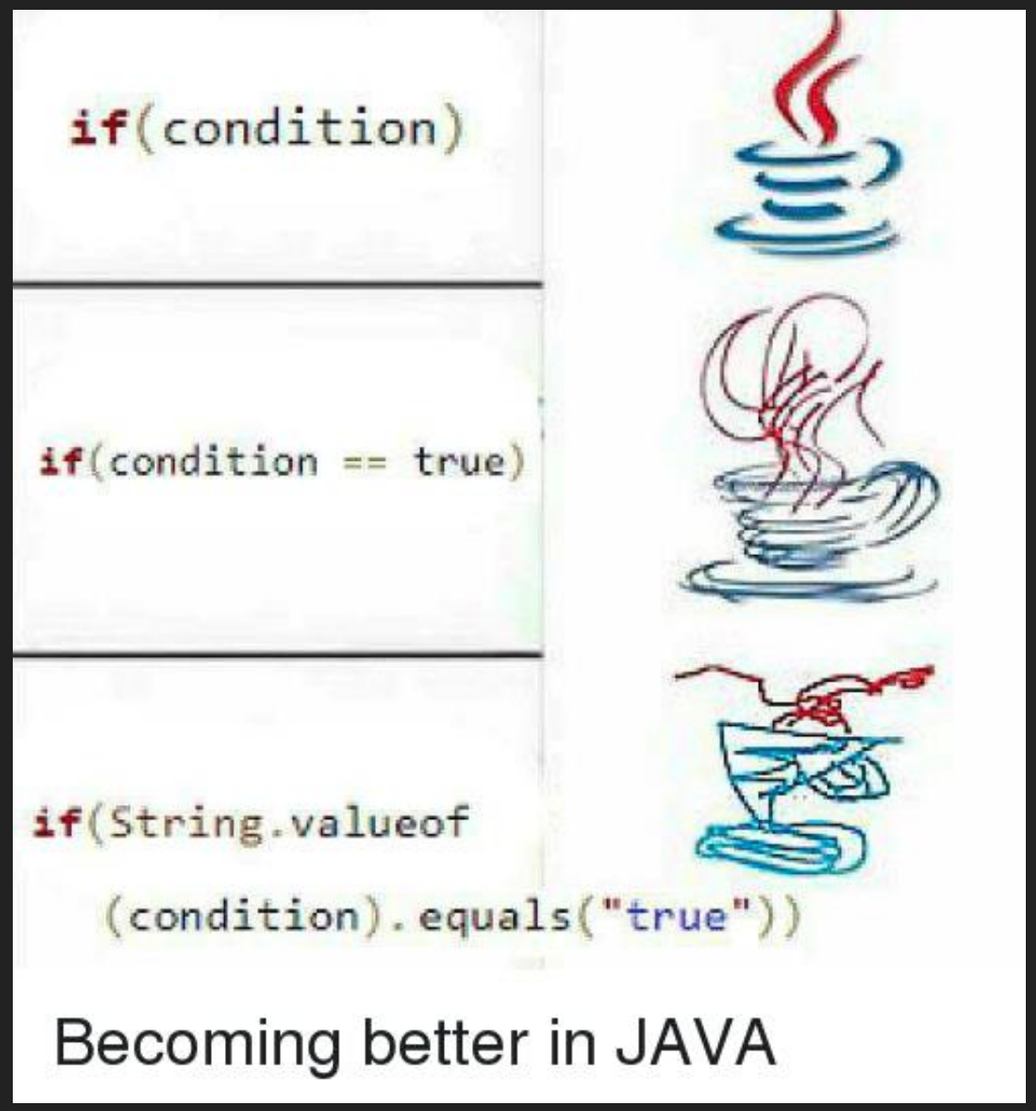
Assignment Operators

+=, -=, *=, /=

Bitwise Operators

Biasanya dipakai ole kalangan pembuat algorithma encryptor

CONDITIONS

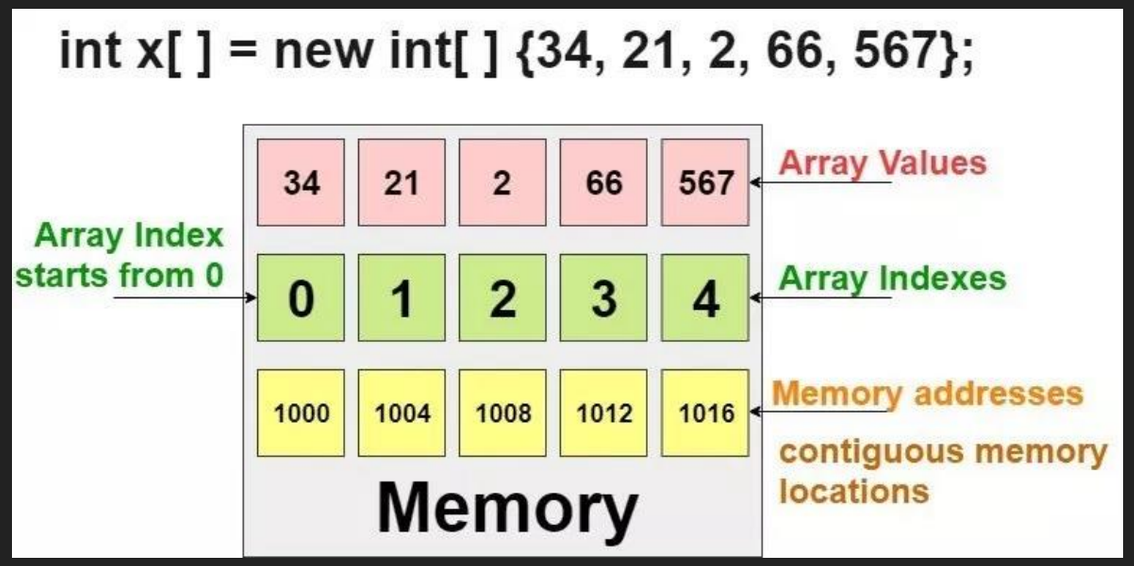


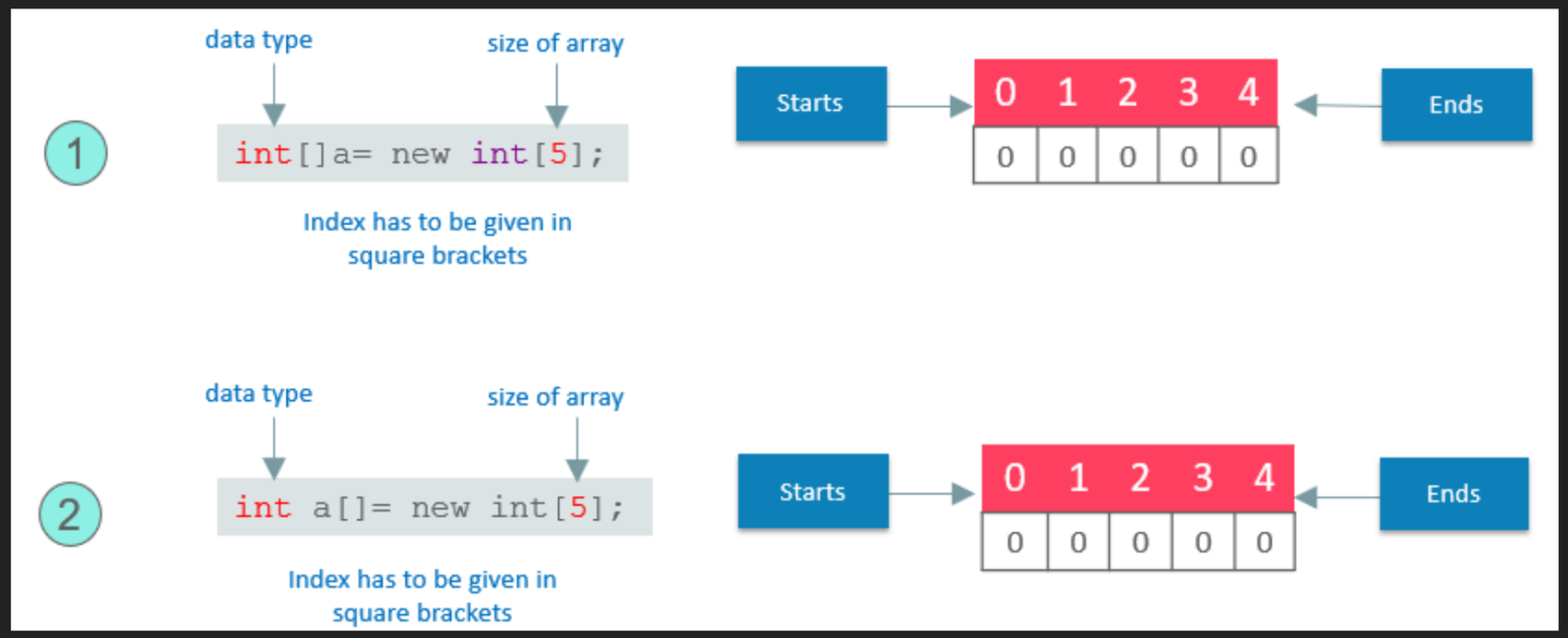
There are two types of people:

```
if (Condition) {
  Statement
  /* ....
   */
}
```

```
if (Condition)
{
  Statement
  /* ....
   */
}
```

ARRAY


int x[ ] = new int[ ] {34, 21, 2, 66, 567};
34
21
2
66
567
Array Values
Array Index starts from 0
0
1
2
3
4
Array Indexes
1000
1004
1008
1012
1016
Memory addresses
contiguous memory locations
Memory


1
data type
size of array
int[]a= new int[5];
Index has to be given in square brackets
Starts
0 1 2 3 4
0 0 0 0 0
Ends
2
data type
size of array
int a[]= new int[5];
Index has to be given in square brackets
Starts
0 1 2 3 4
0 0 0 0 0
Ends

LOOP

Java hanya memiliki 3 tipe Loop

- For
- While
- Do-while

LOOP

Initial Condition Iteration

```
for(int i=0;i<=10;i++){
    System.out.println(i);
}
```

Apa kah fungsi 2 statement ini?

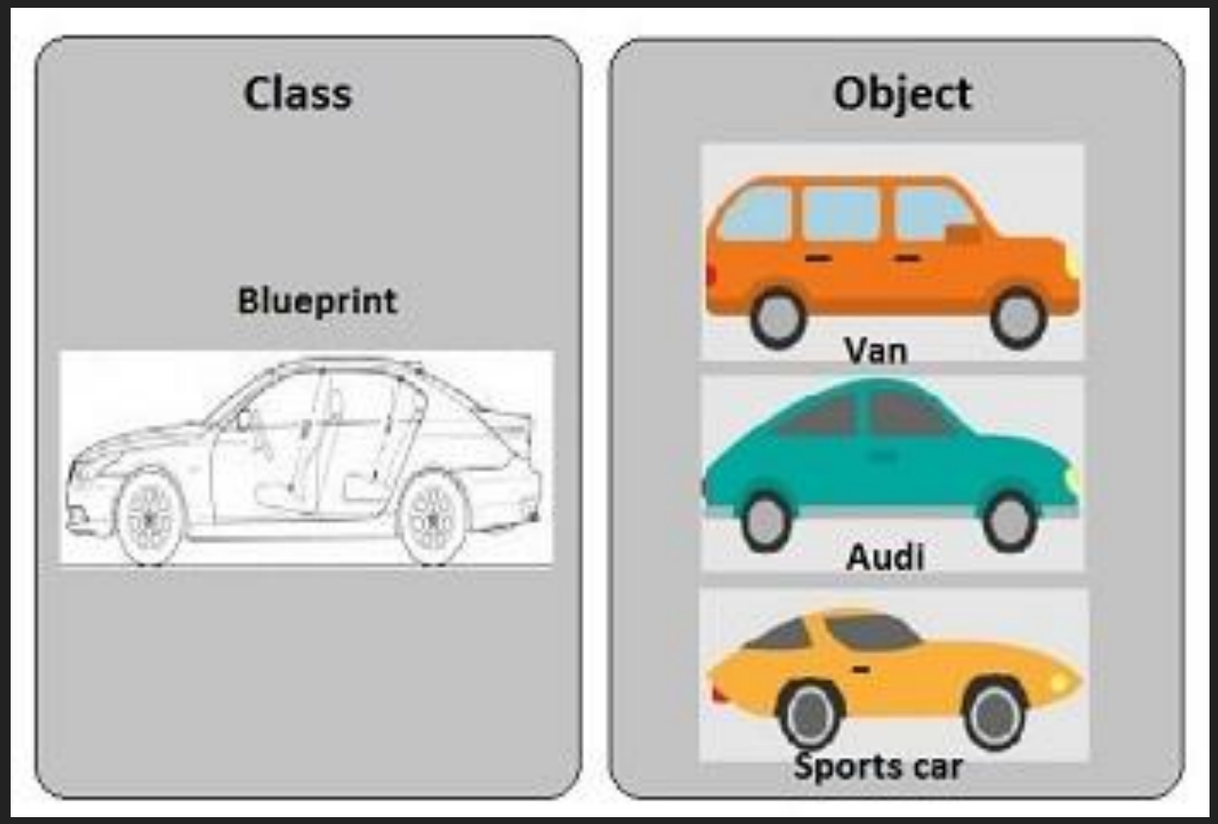
```
do {
    BufferedReader reader = new BufferedReader( new InputStreamReader(System.in));
    nama = reader.readLine();
}while(nama.equals("angga"));
```

- Break
- Continue

```
while(nama.equals("angga")){
    BufferedReader reader = new BufferedReader( new InputStreamReader(System.in));
    nama = reader.readLine();
}
```

OBJECT ORIENTED PROGRAMM

CLASS TO AN OBJECT



THE KEYWORD IS “BLUEPRINT”

Angga Raditya

CLASS, OBJECT, CONSTRUCTOR

```java
public class Car {

    private String colour;
    public Boolean isStart;
    private Integer fuel;

    public Car(String colour){
        this.colour = colour;
        this.fuel = 0;
    }

    public void fillFuel(int fuel){
        this.fuel = this.fuel + fuel;
    }

    public void engineStart(){
        if(this.fuel>0){
            System.out.println("Brum brum");
        } else {
            System.out.println("Insufficient fuel");
        }
    }

    public String print() {
        return "Car{" +
                "colour='" + colour + '\'' +
                ", isStart=" + isStart +
                ", fuel=" + fuel +
                '}';
    }
}
```

```java
import com.enigma.model.Car;
import java.io.IOException;

public class Main {

    public static void main(String[] args) throws IOException {

        Car toyotaInova = new Car("Yellow");
        toyotaInova.engineStart();

    }
}
```

```
➜  src javac Main.java
➜  src java Main
Insufficient fuel
➜  src
```

ACCESS MODIFIER

- Public
- Private
- Protected
- Final
- Static

Should choose only 1

Class:

- Cannot be private
- Cannot be protected
- Should be public
- It can be final, but……

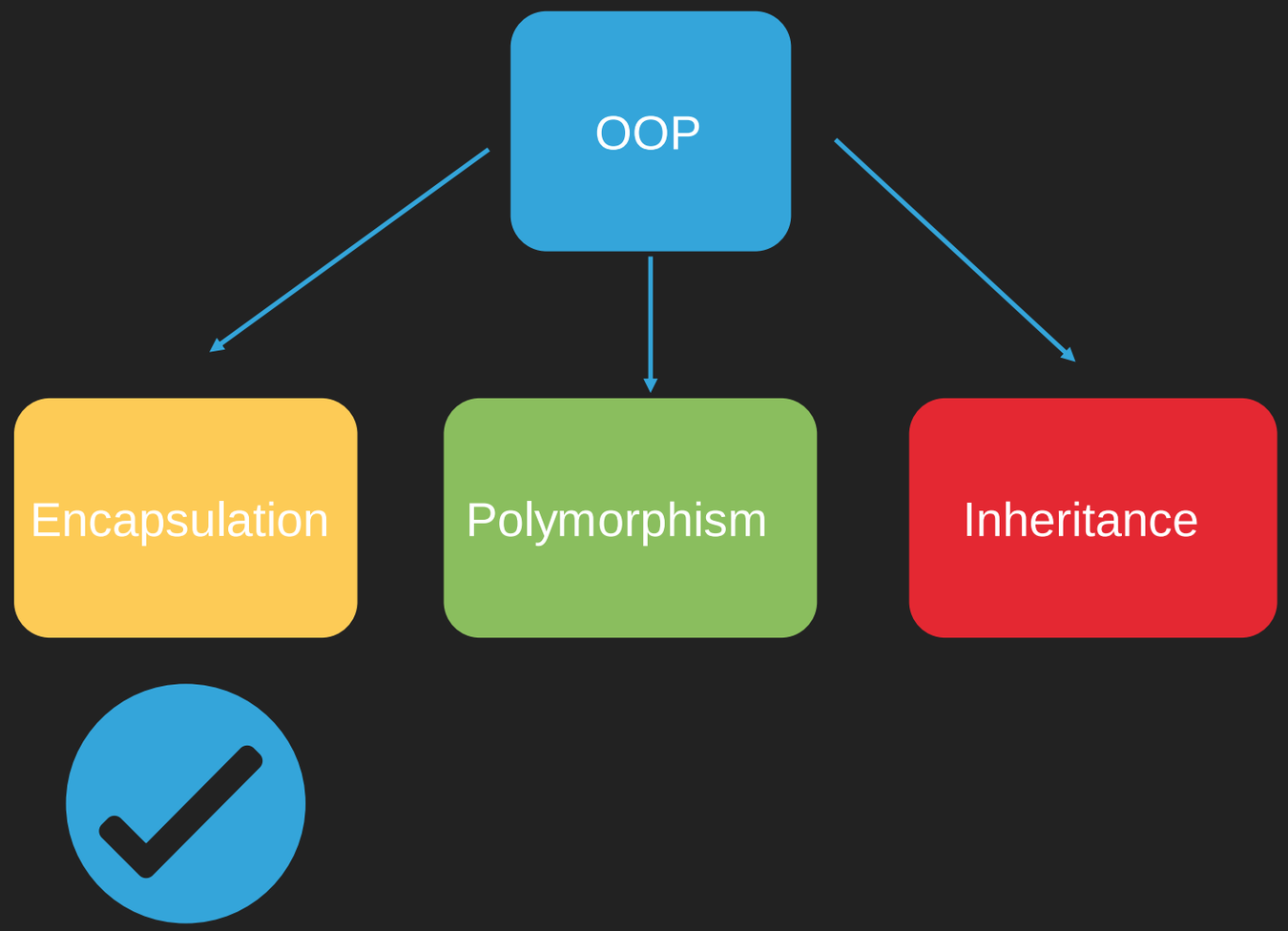
Attribute:

- It can be public
- Can be private
- Can be protected
- It can be final, but need to initialised
- Can be static, if ….

Method:

All of them, but of course you should choose between public, private and protected

OOP



ENCAPSULATION

```java
public class Car {

    private String colour;
    public Boolean isStart;
    private Integer fuel;

    public Car(String colour){
        this.colour = colour;
        this.fuel = 0;
    }

    public void fillFuel(int fuel){
        this.fuel = this.fuel + fuel;
    }

    public void engineStart(){
        if(this.fuel>0){
            System.out.println("Brum brum");
        } else {
            System.out.println("Insufficient fuel");
        }
    }

    public String print() {
        return "Car{" +
                "colour='" + colour + '\'' +
                ", isStart=" + isStart +
                ", fuel=" + fuel +
                '}';
    }
}
```

Watch

```java
public static void main(String[] args) throws IOException {

    Car toyotaInova = new Car("Yellow");
    toyotaInova.isStart=true;
    System.out.println(toyotaInova.print());

}
```

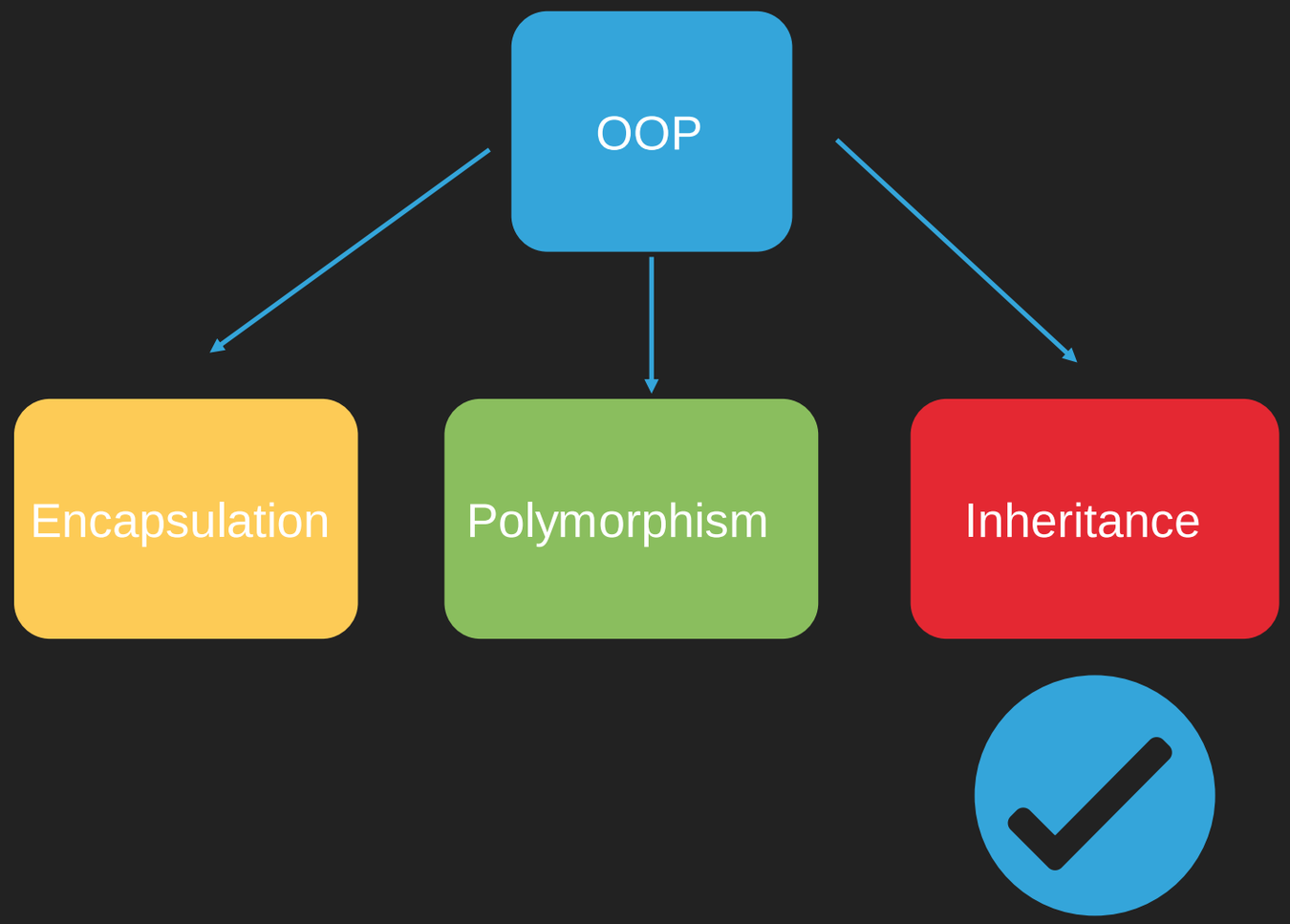
Direct access

```
➜  src javac Main.java
➜  src java Main
Car{colour='Yellow', isStart=true, fuel=0}
➜  src
```

The funny part, engine start but no fuel. It doesn't make sense. It happens because the direct access attribute

ENCAPSULATION

OOP



INHERITANCE

```java
public class Rectangle {

    protected Double length;
    protected Double width;


    public Rectangle(Double length, Double width){
        this.length = length;
        this.width = width;
    }

    public Double getSurface(){
        return length*width;
    }


    Double getRound(){
        return 2*(length+width);
    }

    public String print() {
        return "Rectangle{" +
                "length=" + length +
                ", width=" + width +
                ", round="+ getRound() +
                ", surface="+ getSurface() +
                '}';
    }
}
```

```java
public class Block extends Rectangle {

    private Double height;

    public Double getVolume(){
        return this.length*this.width*this.height;
    }

    public Block(Double length, Double width, Double height){
        super(length,width);
        this.height=height;
    }

    @Override
    public Double getSurface(){
        return 2*(this.length*this.width)+
                2*(this.length*this.height)+
                2*(this.width*this.height);
    }


    public String print() {
        return "Block{ length="+this.length+
                ", width="+this.width+
                ", height="+this.height+
                ", surface="+getSurface()+
                ", volume="+getRound()+"}";
    }
}
```

INHERITANCE

- It will inherit attribute
- It inherit method
- The method can be override
- You can do Parent object = new Child();

OBJECT INTERACTION

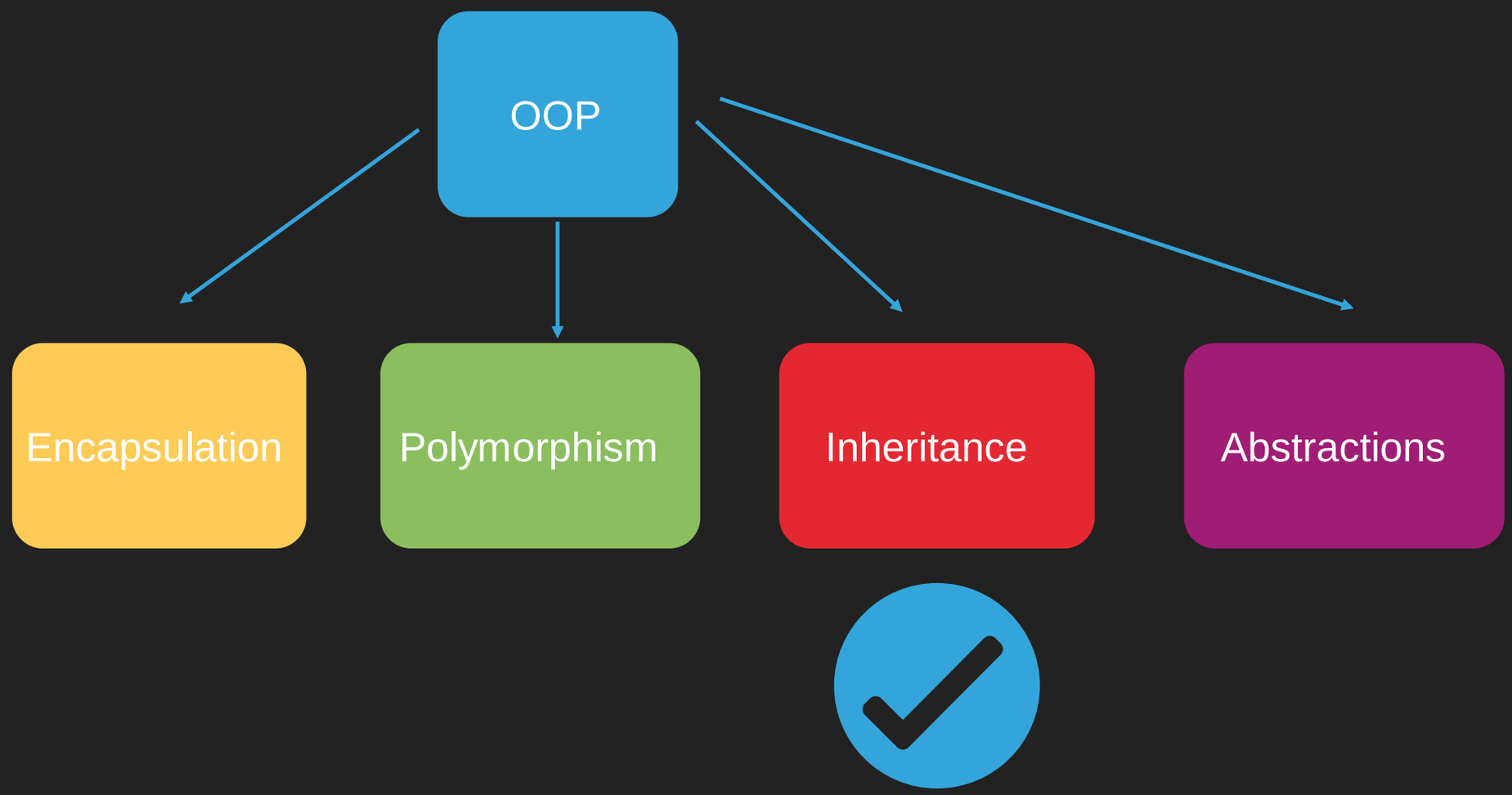
```java
public class Heroes {

    private String name;
    private Integer hp;
    private Integer mana;
    private Integer baseDamage;
    private Skill skill1;

    public Heroes(Integer hp, Integer mana, Integer baseDamage) {
        this.hp = hp;
        this.mana = mana;
        this.baseDamage = baseDamage;
    }
    public void attack(Heroes heroes) {
        heroes.getHit(this.baseDamage);
    }
    public void getHit(Integer damage) {
        this.hp =  this.hp-damage;
    }
    public String print() {
        return "Heroes{" +
                "hp=" + hp +
                ", mana=" + mana +
                ", baseDamage=" + baseDamage +
                '}';
    }
}
```

```java
public class Main {

    public static void main(String[] args) throws IOException {

        Heroes gatotKaca = new Heroes(1000,500, 20);
        Heroes saitama = new Heroes(1000,500,50);

        saitama.attack(gatotKaca);

        System.out.println(gatotKaca.print());
        System.out.println(saitama.print());
    }


}
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_202.j
Heroes{hp=950, mana=500, baseDamage=20}
Heroes{hp=1000, mana=500, baseDamage=50}

Process finished with exit code 0
```

TEXT

OOP


OOP
Encapsulation
Polymorphism
Inheritance
Abstractions

POLYMORPHISM

```
public class Heroes {

    public String name;
    protected Integer hp;
    protected Integer mana;
    protected Integer baseDamage;
    protected Integer attackSpeed;

    public Heroes(String name, Integer hp, Integer mana, Integer baseDamage, Integer attackSpeed) {
        this.name = name;
        this.hp = hp;
        this.mana = mana;
        this.baseDamage = baseDamage;
        this.attackSpeed = attackSpeed;
    }
    public void attack(Heroes heroes) {
        heroes.getHit(this.baseDamage);
    }
    public void getHit(Integer damage) {
        this.hp =  this.hp-damage;
    }
    public String print() {
        return "Heroes{" +
                "hp=" + hp +
                ", mana=" + mana +
                ", baseDamage=" + baseDamage +
                '}';
    }
}
```

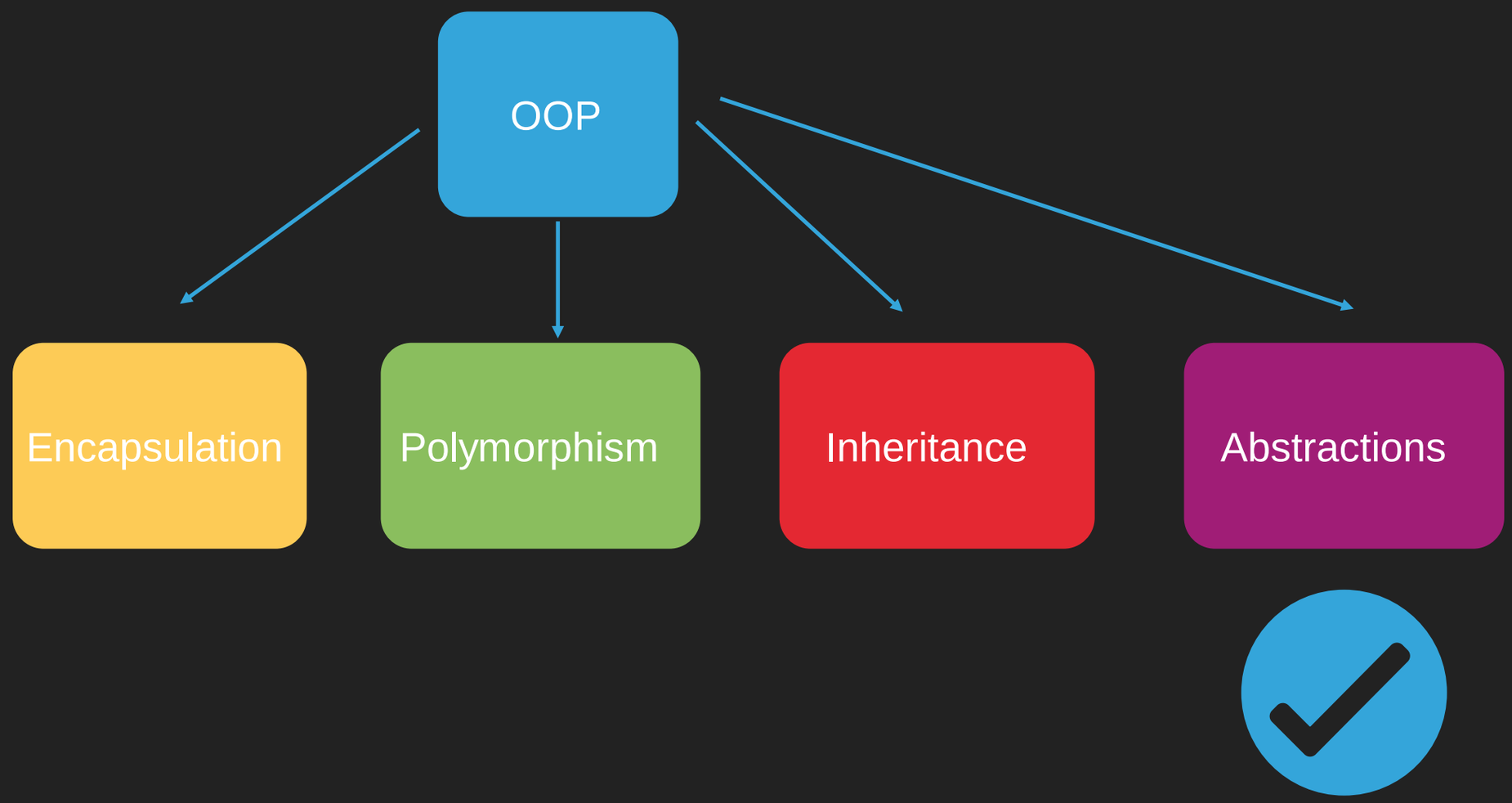
POLYMORPHISM

```
public class TankerHeroes extends Heroes{

    public TankerHeroes(String name, Integer hp, Integer mana, Integer damage, Integer attackspeed){
        super(name,hp,mana,damage, attackspeed);
        this.hp = this.hp + 300;
    }
}
```

```
public class AssasinHeroes extends Heroes {

    public AssasinHeroes(String name, Integer hp, Integer mana, Integer damage, Integer attackSpeed){
        super(name, hp,mana,damage, attackSpeed);
        this.attackSpeed = this.attackSpeed +100;
    }

}
```

```
public class MageHero extends Heroes{
    public MageHero(String name, Integer hp, Integer mana, Integer damage, Integer attackSpeed){
        super(name, hp,mana,damage, attackSpeed);
        this.mana = this.mana + 300;
    }
}
```

OOP


OOP
Encapsulation
Polymorphism
Inheritance
Abstractions

ABSTRACTIONS

```java
public abstract class Shape {

    abstract Double getRound();
    abstract Double getSurface();

}
```

```java
public class Rectangle extends Shape{

    protected Double length;
    protected Double width;


    public Rectangle(Double length, Double width){
        this.length = length;
        this.width = width;
    }

    @Override
    public Double getSurface(){
        return length*width;
    }

    @Override
    Double getRound(){
        return 2*(length+width);
    }

    public String print() {
        return "Rectangle{" +
                "length=" + length +
                ", width=" + width +
                ", round="+ getRound() +
                ", surface="+ getSurface() +
                '}';
    }
}
```

```java
public class Circle extends Shape {

    public Integer r;
    private final Double pi=3.14;

    @Override
    public Double getSurface(){
        Double surface = pi*r*r;
        return surface;
    }

    @Override
    public Double getRound(){
        Double round = pi*r*2;
        return round;
    }


    public Double getHalfSurface(){
        return  getSurface()/2;
    }

    public Double getHalfRound(){
        return  getRound()/2;
    }

    public String print() {
        return "Circle{" + "r=" + r +
                ", pi=" + pi +
                ", surface = "+getSurface()+
                ", round = "+getRound()+'}';
    }
}
```

ABSTRACTIONS

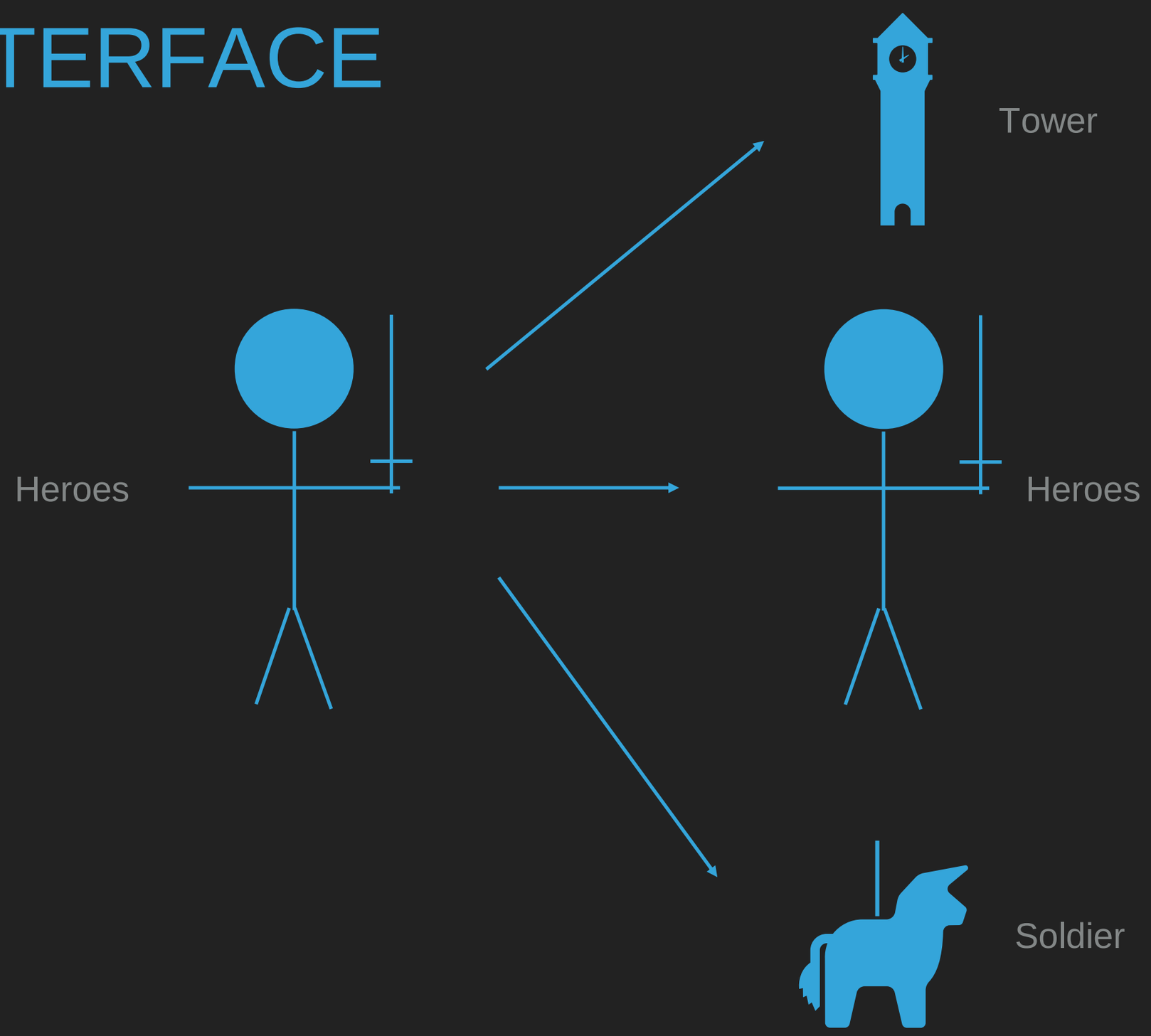
- Must be declare with abstract keywords
- It can have abstract or non abstract method
- Cannot be create direct object

INTERFACE

```
public class Heroes {

    public String name;
    protected Integer hp;
    protected Integer mana;
    protected Integer baseDamage;
    protected Integer attackSpeed;

    public Heroes(String name, Integer hp, Integer mana, Integer baseDamage,
Integer attackSpeed) {
        this.name = name;
        this.hp = hp;
        this.mana = mana;
        this.baseDamage = baseDamage;
        this.attackSpeed = attackSpeed;
    }
    public void attack(Heroes heroes) {
        heroes.getHit(this.baseDamage);
    }
    public void getHit(Integer damage) {
        this.hp =  this.hp-damage;
    }
    public String print() {
        return "Heroes{" +
                "hp=" + hp +
                ", mana=" + mana +
                ", baseDamage=" + baseDamage +
                '}';
    }
}
```

It can hit only a Hero

INTERFACE



INTERFACE

```java
public interface HitAble {

    public void getHit(Integer damage);

}
```

Now it can hit whatever is hit able

```java
public class Heroes implements HitAble {

    public String name;
    protected Integer hp;
    protected Integer mana;
    protected Integer baseDamage;
    protected Integer attackSpeed;

    public Heroes(String name, Integer hp, Integer mana, Integer baseDamage, Integer
attackSpeed) {
        this.name = name;
        this.hp = hp;
        this.mana = mana;
        this.baseDamage = baseDamage;
        this.attackSpeed = attackSpeed;
    }
    public void attack(HitAble hitAble) {
        hitAble.getHit(this.baseDamage);
    }

    @Override
    public void getHit(Integer damage) {
        this.hp =  this.hp-damage;
    }
    public String print() {
        return "Heroes{" +
                "hp=" + hp +
                ", mana=" + mana +
                ", baseDamage=" + baseDamage +
                '}';
    }
}
```

INTERFACE

```java
public class Tower implements HitAble{
    Integer hp;

    public Tower(Integer hp){
        this.hp = hp;
    }

    @Override
    public void getHit(Integer damage) {
        this.hp = this.hp - damage;
    }

    public String print() {
        return "Tower{" +
                "hp=" + hp +
                '}';
    }
}
```

```java
public class Main {

    public static void main(String[] args) throws IOException {

        Heroes gatotKaca = new Heroes("Gatot Kaca",1000,500, 20, 10);
        Heroes saitama = new Heroes("Saitama",1000,500, 20, 10);
        Tower tower = new Tower(10000);

        saitama.attack(tower);

        System.out.println(gatotKaca.print());
        System.out.println(saitama.print());
        System.out.println(tower.print());
    }


}
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_202.j
Heroes{hp=1000, mana=500, baseDamage=20}
Heroes{hp=1000, mana=500, baseDamage=20}
Tower{hp=9980}

Process finished with exit code 0
```

OBJECT VALUE - EQUALS & HASHCODE

IT WILL BE EASIER USING COLLECTIONS FRAMEWORKS, WHEN WE KNOW HOW THE OBJECT WORKS

Angga Raditya

OBJECT VALUE

```java
public class Circle extends Shape {

    private Integer r;
    private final Double pi=3.14;

    public Circle(Integer r) {
        this.r = r;
    }

    @Override
    public Double getSurface(){
        Double surface = pi*r*r;
        return surface;
    }

    @Override
    public Double getRound(){
        Double round = pi*r*2;
        return round;
    }


    public String print() {
        return "Circle{" + "r=" + r +
                ", pi=" + pi +
                ", surface = "+getSurface()+
                ", Class = "+getClass()+
                ", round = "+getRound()+'}';
    }
}
```

```java
public static void main(String[] args) throws IOException, InterruptedException {

    Circle object = new Circle(10);
    System.out.println(object.equals(new Circle(10)));

}
```

```
false

Process finished with exit code 0
```

OBJECT VALUE - EQUALS & HASHCODE

```java
package java.lang;

/**
 * Class {@code Object} is the root of the class hierarchy.
 * Every class has {@code Object} as a superclass. All objects,
 * including arrays, implement the methods of this class.
 *
 * @author  unascribed
 * @see     java.lang.Class
 * @since   JDK1.0
 */
public class Object {

    private static native void registerNatives();
    static {
        registerNatives();
    }

    /**...*/
    @Contract(pure=true) public final native Class<?> getClass();

    /**...*/
    public native int hashCode();

    /**...*/
    public boolean equals(Object obj) {
        return (this == obj);
    }

    /**...*/
    protected native Object clone() throws CloneNotSupportedException;
```

It only compare the address

WE MUST DEFINE WHAT VALUES MAKE AN OBJECT UNIQUE

Angga Raditya

OBJECT VALUE - EQUALS & HASHCODE

```java
public class Circle extends Shape {

    private Integer r;
    private final Double pi=3.14;

    public Circle(Integer r) {
        this.r = r;
    }

    @Override
    public Double getSurface(){
        Double surface = pi*r*r;
        return surface;
    }

    @Override
    public Double getRound(){
        Double round = pi*r*2;
        return round;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Circle circle = (Circle) o;
        return Objects.equals(r, circle.r);
    }


    public String print() {
        return "Circle{" + "r=" + r +
                ", pi=" + pi +
                ", surface = "+getSurface()+
                ", Class = "+getClass()+
                ", round = "+getRound()+'}';
    }
}
```

```java
public class Main {

    public static void main(String[] args) throws IOException, InterruptedException {
        Circle object = new Circle(10);
        System.out.println(object.equals(new Circle(10)));
    }

}
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_202.jdk/Contents/Home/bin/java ...
true

Process finished with exit code 0
```

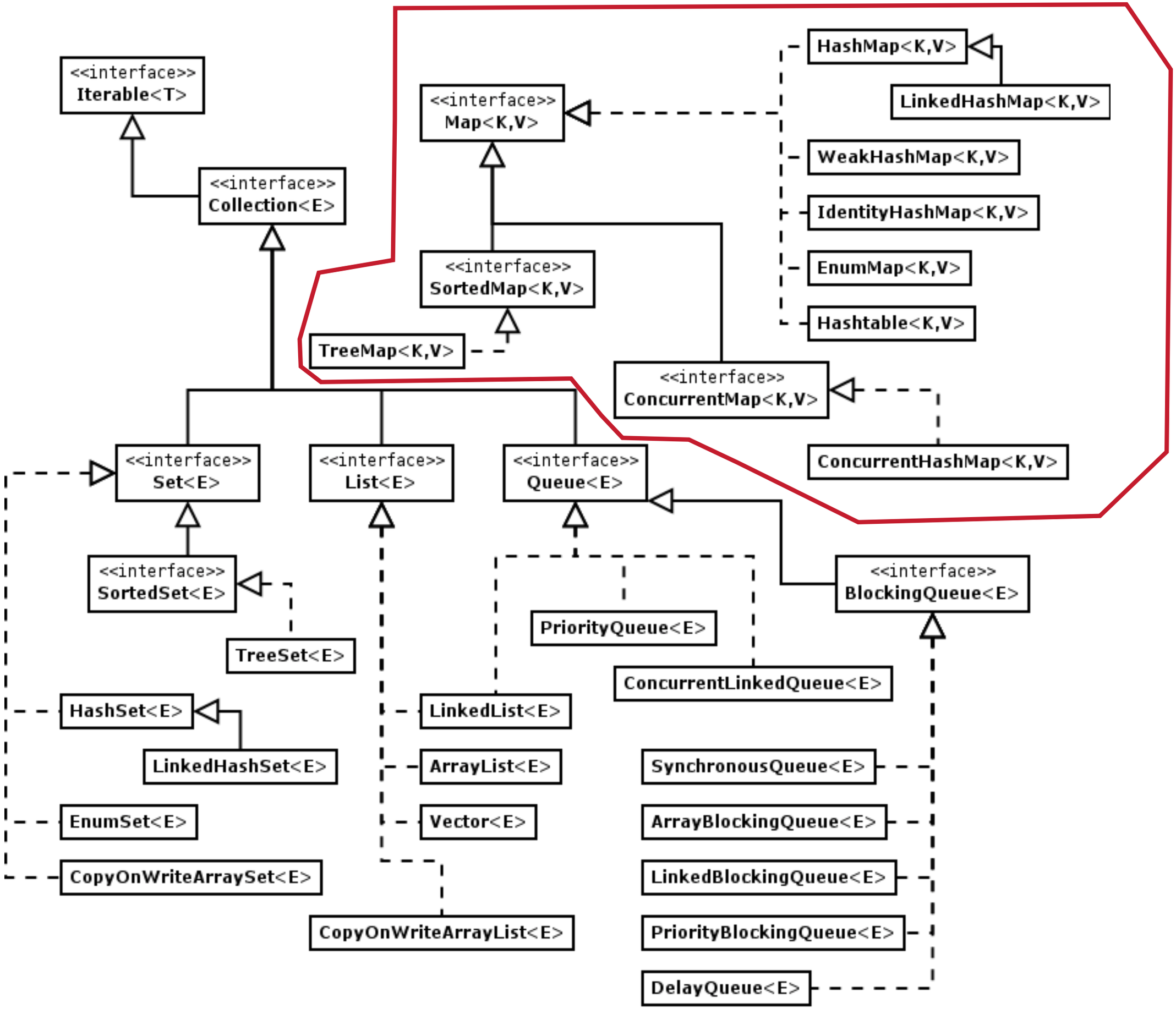
Now it returns the “r” comparison. “r" became the unique value of circle

May be in this case using “r” as a unique value is less relevant. But it will start to make sense when we using Collections Framework

COLLECTIONS

COLLECTIONS

- The Interfaces: Set, List, Queue, Deque
- The Class : Arraylist, Vector, LinkedList, HashSet, etc…


<<interface>>
Iterable<T>
<<interface>>
Collection<E>
<<interface>>
Map<K,V>
HashMap<K,V>
LinkedHashMap<K,V>
WeakHashMap<K,V>
IdentityHashMap<K,V>
EnumMap<K,V>
Hashtable<K,V>
<<interface>>
SortedMap<K,V>
TreeMap<K,V>
<<interface>>
ConcurrentMap<K,V>
ConcurrentHashMap<K,V>
<<interface>>
Set<E>
<<interface>>
List<E>
<<interface>>
Queue<E>
<<interface>>
SortedSet<E>
TreeSet<E>
HashSet<E>
LinkedHashSet<E>
EnumSet<E>
CopyOnWriteArraySet<E>
LinkedList<E>
ArrayList<E>
Vector<E>
CopyOnWriteArrayList<E>
PriorityQueue<E>
ConcurrentLinkedQueue<E>
<<interface>>
BlockingQueue<E>
SynchronousQueue<E>
ArrayBlockingQueue<E>
LinkedBlockingQueue<E>
PriorityBlockingQueue<E>
DelayQueue<E>

COLLECTIONS: SET-HASHSET

SET - HASHSET USING VALUE DATA TYPE

```java
public class Main {
    public static void main(String[] args) throws IOException, InterruptedException {
        Set<Integer> integerSet = new HashSet<>();

        integerSet.add(10);
        integerSet.add(5);
        integerSet.add(1);
        integerSet.add(1);
        integerSet.add(1);
        integerSet.add(3);
        integerSet.add(4);

        for (Integer nilai: integerSet) {
            System.out.println(nilai);
        }
        System.out.println("Size:"+integerSet.size());
    }
}
```

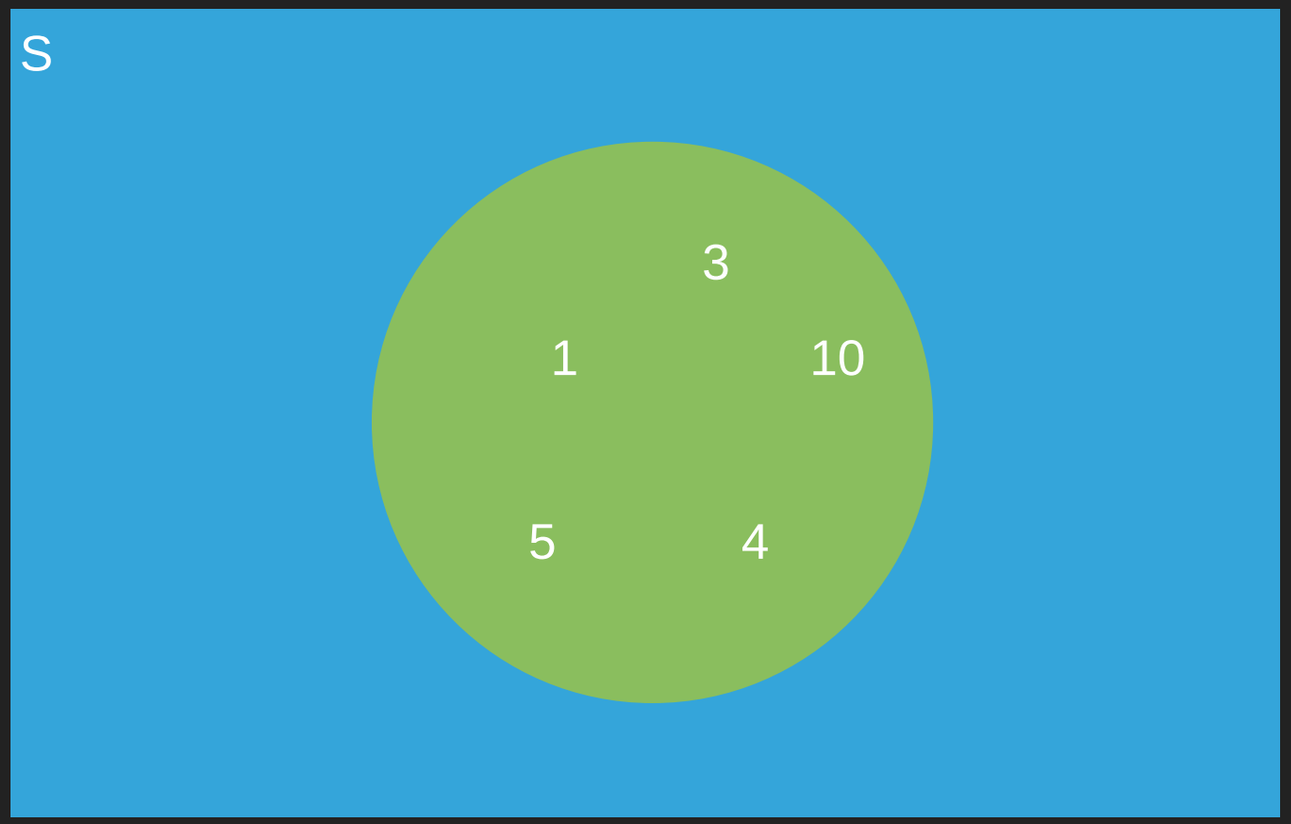
```
/Library/Java/JavaVirtualMachines/jdk1.8.0_202.jdk/Contents/Home/bin/java ...
1
3
4
5
10
Size:5

Process finished with exit code 0
```

Where the others “1” ?????

SET IN BAHASA MEANS “HIMPUNAN”

Angga Raditya

SET - HASHSET



SET - HASHSET

- Each element in set is not indexed, obviously because order is unnecessary in set.
- It contains unique element only
- It will use the "HashCode()" as unique value indicator
- It can store null values;

SET - HASHSET USING OBJECT

```java
public class Circle extends Shape {

    private Integer r;
    private final Double pi=3.14;

    public Circle(Integer r) {
        this.r = r;
    }

    @Override
    public Double getSurface(){
        Double surface = pi*r*r;
        return surface;
    }

    @Override
    public Double getRound(){
        Double round = pi*r*2;
        return round;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Circle circle = (Circle) o;
        return Objects.equals(r, circle.r);
    }


    public String print() {
        return "Circle{" + "r=" + r +
                ", pi=" + pi +
                ", surface = "+getSurface()+
                ", Class = "+getClass()+
                ", round = "+getRound()+'}';
    }
}
```

```java
public class Main {

    public static void main(String[] args) throws IOException, InterruptedException {
        Set<Circle> circleSet = new HashSet<>();

        circleSet.add(new Circle(10));
        circleSet.add(new Circle(5));
        circleSet.add(new Circle(1));
        circleSet.add(new Circle(1));
        circleSet.add(new Circle(1));
        circleSet.add(new Circle(3));
        circleSet.add(new Circle(4));

        for (Circle circle: circleSet) {
            System.out.println(circle.print());
        }
        System.out.println("Size:"+circleSet.size());

    }

}
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_202.jdk/Contents/Home/bin/java ...
Circle{r=10, pi=3.14, surface = 314.0, Class = class com.enigma.model.Circle, round = 62.80000000000004
Circle{r=1, pi=3.14, surface = 3.14, Class = class com.enigma.model.Circle, round = 6.28}
Circle{r=5, pi=3.14, surface = 78.5, Class = class com.enigma.model.Circle, round = 31.40000000000002}
Circle{r=1, pi=3.14, surface = 3.14, Class = class com.enigma.model.Circle, round = 6.28}
Circle{r=4, pi=3.14, surface = 50.24, Class = class com.enigma.model.Circle, round = 25.12}
Circle{r=1, pi=3.14, surface = 3.14, Class = class com.enigma.model.Circle, round = 6.28}
Circle{r=3, pi=3.14, surface = 28.259999999999998, Class = class com.enigma.model.Circle, round = 18.84}
Size:7
```

Duplications still occurs

SET - HASHSET USING OBJECT

```java
public class Circle extends Shape {

    private Integer r;
    private final Double pi=3.14;

    public Circle(Integer r) {
        this.r = r;
    }

    @Override
    public Double getSurface(){
        Double surface = pi*r*r;
        return surface;
    }

    @Override
    public Double getRound(){
        Double round = pi*r*2;
        return round;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Circle circle = (Circle) o;
        return Objects.equals(r, circle.r);
    }

    @Override
    public int hashCode() {
        return Objects.hash(r);
    }

    public String print() {
        return "Circle{" + "r=" + r +
                ", pi=" + pi +
                ", surface = "+getSurface()+
                ", Class = "+getClass()+
                ", round = "+getRound()+'}';
    }
}
```

Add hashcode()

```java
public class Main {

    public static void main(String[] args) throws IOException, InterruptedException {
        Set<Circle> circleSet = new HashSet<>();

        circleSet.add(new Circle(10));
        circleSet.add(new Circle(5));
        circleSet.add(new Circle(1));
        circleSet.add(new Circle(1));
        circleSet.add(new Circle(1));
        circleSet.add(new Circle(3));
        circleSet.add(new Circle(4));

        for (Circle circle: circleSet) {
            System.out.println(circle.print());
        }
        System.out.println("Size:"+circleSet.size());


    }

}
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_202.jdk/Contents/Home/bin/java ...
Circle{r=1, pi=3.14, surface = 3.14, Class = class com.enigma.model.Circle, round
Circle{r=3, pi=3.14, surface = 28.259999999999998, Class = class com.enigma.model
Circle{r=4, pi=3.14, surface = 50.24, Class = class com.enigma.model.Circle, roun
Circle{r=5, pi=3.14, surface = 78.5, Class = class com.enigma.model.Circle, round
Circle{r=10, pi=3.14, surface = 314.0, Class = class com.enigma.model.Circle, rou
Size:5

Process finished with exit code 0
```

The duplications now gone

NOW YOU KNOW WHAT HASHCODE IS?

Angga Raditya

COLLECTIONS: LIST-ARRAYLIST

LIST - ARRAY LIST

```java
public class Main {

    public static void main(String[] args) throws IOException, InterruptedException {
        List<Integer> integerList = new ArrayList<>();

        integerList.add(10);
        integerList.add(5);
        integerList.add(1);
        integerList.add(1);
        integerList.add(1);
        integerList.add(3);
        integerList.add(4);

        for (Integer nilai: integerList) {
            System.out.println(nilai);
        }
        System.out.println("Size:"+integerList.size());
    }
}
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_202.jdk/Contents/Home/bin/java ...
10
5
1
1
1
3
4
Size:7
```

Duplications

LIST - ARRAYLIST

- Element indexed.
- It can contains duplicate element
- It will use the “equals()” as unique value indicator

LIST-ARRAYLIST

```java
public class Circle extends Shape {

    private Integer r;
    private final Double pi=3.14;

    public Circle(Integer r) {
        this.r = r;
    }

    @Override
    public Double getSurface(){
        Double surface = pi*r*r;
        return surface;
    }
    @Override
    public Double getRound(){
        Double round = pi*r*2;
        return round;
    }
    @Override
    public int hashCode() {
        return Objects.hash(r);
    }
    public String print() {
        return "Circle{" + "r=" + r +
                ", pi=" + pi +
                ", surface = "+getSurface()+
                ", Class = "+getClass()+
                ", round = "+getRound()+'}';
    }
}
```

```java
public class Main {

    public static void main(String[] args) throws IOException, InterruptedException {
        List<Circle> circleList = new ArrayList<>();

        circleList.add(new Circle(10));
        circleList.add(new Circle(5));
        circleList.add(new Circle(1));
        circleList.add(new Circle(1));
        circleList.add(new Circle(1));
        circleList.add(new Circle(3));
        circleList.add(new Circle(4));


        for (Circle circle: circleList) {
            System.out.println(circle.print());
        }
        System.out.println("Size:"+circleList.size());

        Circle findRings = new Circle(3);
        System.out.println("Contains :"+circleList.contains(findRings));

    }
}
```

Try to find circles with r=3

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_202.jdk/Contents/Home/bin/java .
Circle{r=10, pi=3.14, surface = 314.0, Class = class com.enigma.model.Circl
Circle{r=5, pi=3.14, surface = 78.5, Class = class com.enigma.model.Circle,
Circle{r=1, pi=3.14, surface = 3.14, Class = class com.enigma.model.Circle,
Circle{r=1, pi=3.14, surface = 3.14, Class = class com.enigma.model.Circle,
Circle{r=1, pi=3.14, surface = 3.14, Class = class com.enigma.model.Circle,
Circle{r=3, pi=3.14, surface = 28.259999999999998, Class = class com.enigma
Circle{r=4, pi=3.14, surface = 50.24, Class = class com.enigma.model.Circle
Size:7
Contains :false

Process finished with exit code 0
```

Not Found

TEXT

```java
public class Circle extends Shape {

    private Integer r;
    private final Double pi=3.14;

    public Circle(Integer r) {
        this.r = r;
    }

    @Override
    public Double getSurface(){
        Double surface = pi*r*r;
        return surface;
    }

    @Override
    public Double getRound(){
        Double round = pi*r*2;
        return round;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Circle circle = (Circle) o;
        return Objects.equals(r, circle.r);
    }

    @Override
    public int hashCode() {
        return Objects.hash(r);
    }

    public String print() {
        return "Circle{" + "r=" + r +
                ", pi=" + pi +
                ", surface = "+getSurface()+
                ", Class = "+getClass()+
                ", round = "+getRound()+'}';
    }
}
```

```java
public class Main {

    public static void main(String[] args) throws IOException, InterruptedException {
        List<Circle> circleList = new ArrayList<>();

        circleList.add(new Circle(10));
        circleList.add(new Circle(5));
        circleList.add(new Circle(1));
        circleList.add(new Circle(1));
        circleList.add(new Circle(1));
        circleList.add(new Circle(3));
        circleList.add(new Circle(4));


        for (Circle circle: circleList) {
            System.out.println(circle.print());
        }
        System.out.println("Size:"+circleList.size());
        System.out.println("Contains Circle{r=3} :"+circleList.contains(new Circle(3)));
        System.out.println("LastIndexOf Circle{r=1}"+circleList.lastIndexOf(new Circle(1)));

    }
}
```

After we add equals() method

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_202.jdk/Contents/Home/bin/
Circle{r=10, pi=3.14, surface = 314.0, Class = class com.enigma.model
Circle{r=5, pi=3.14, surface = 78.5, Class = class com.enigma.model.(
Circle{r=1, pi=3.14, surface = 3.14, Class = class com.enigma.model.(
Circle{r=1, pi=3.14, surface = 3.14, Class = class com.enigma.model.(
Circle{r=1, pi=3.14, surface = 3.14, Class = class com.enigma.model.(
Circle{r=3, pi=3.14, surface = 28.259999999999998, Class = class com.
Circle{r=4, pi=3.14, surface = 50.24, Class = class com.enigma.model.
Size:7
Contains Circle{r=3} :true
LastIndexOf Circle{r=1}4
```

EXCEPTION

LETS MAKE SOME ERROR

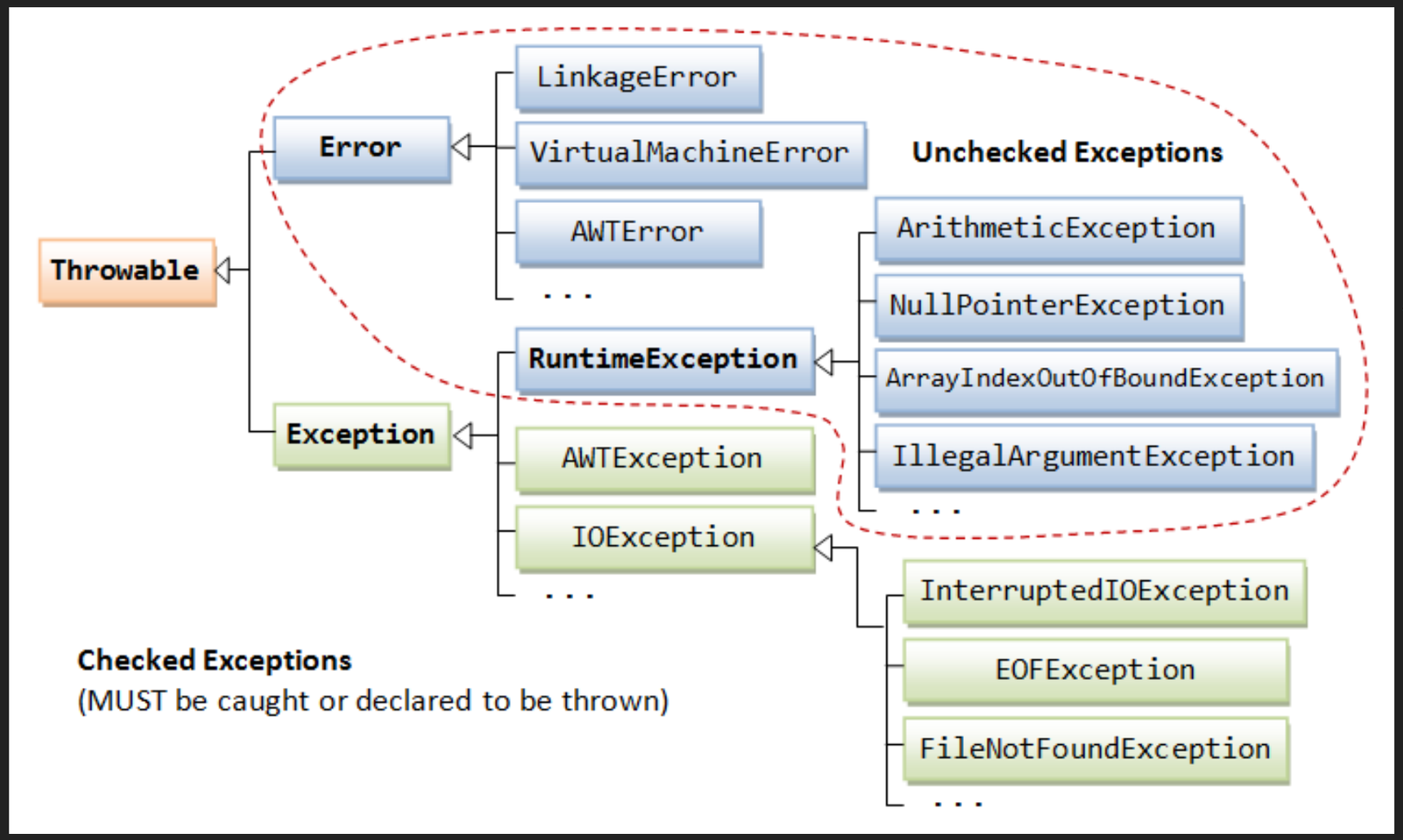
```
public class Main {

    public static void main(String[] args)  {

        int [] setOfNumber = {2,5,6,3};
        System.out.println(setOfNumber[4]);

    }


}
```

IDE seems fine !!

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_202.jdk/Contents/Home/bin/
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:
    at Main.main(Main.java:8)

Process finished with exit code 1
```

This sh*t happens During RunTime

EXCEPTION



Calm down, these is just a few example.

THE WHOLE SH*T IS MUCH MORE

WOULD YOU TRY TO CATCH IT, OR JUST THROW IT? ALWAYS PREPARE FOR THE WORST!!

Angga Raditya

HOW TO HANDLE IT?

```
public static void main(String[] args) throws ArrayIndexOutOfBoundsException {

    int [] setOfNumber = {2,5,6,3};
    System.out.println(setOfNumber[4]);

}
```

Throw it

Hints : when you throw from the main method (psvm) the JVM will catch it

```
public static void main(String[] args)  {

    try{
        int [] setOfNumber = {2,5,6,3};
        System.out.println(setOfNumber[4]);
    }catch (ArrayIndexOutOfBoundsException e){

    }


}
```

Catch it

FILE

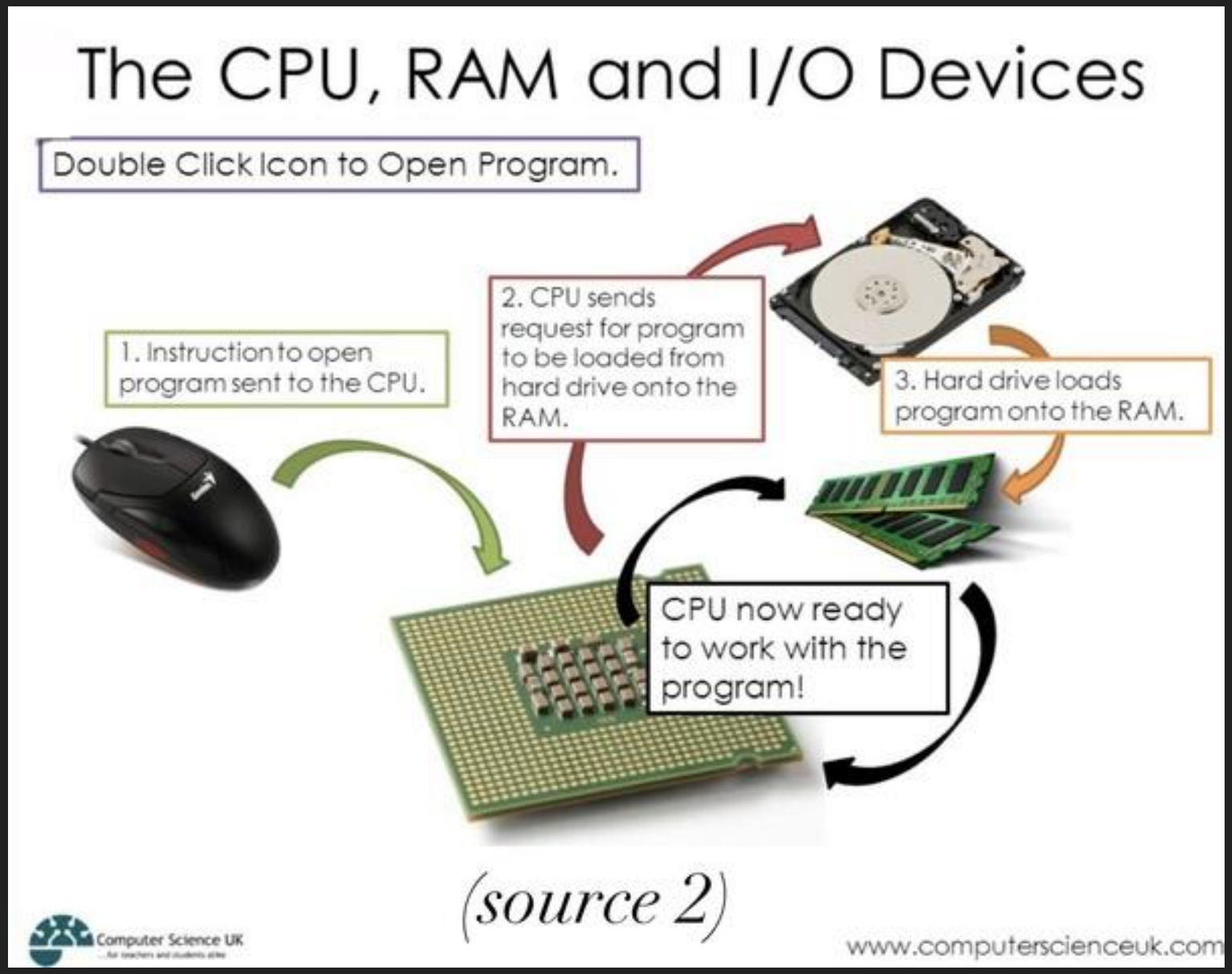
WHAT IF

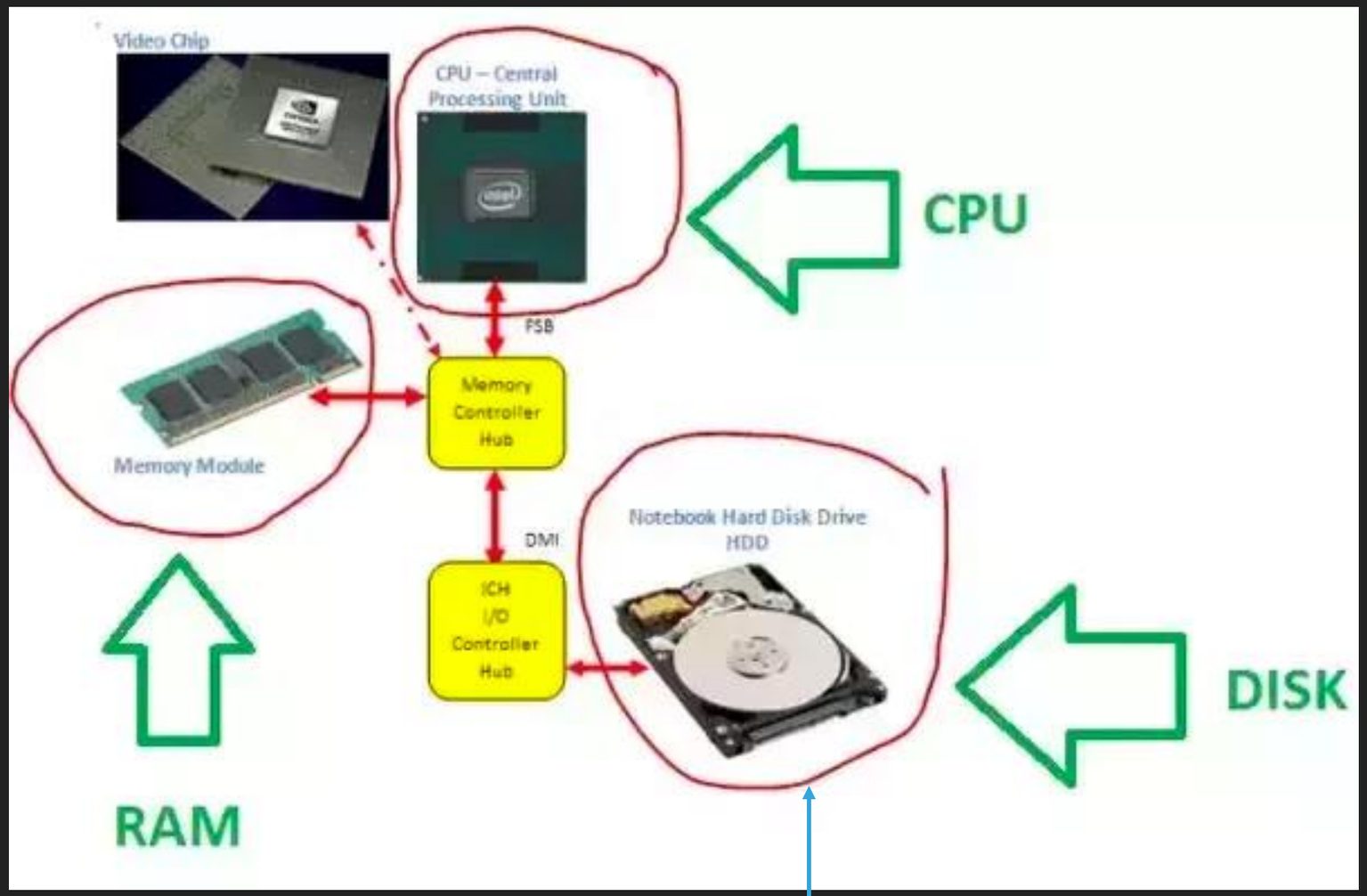
- You want to save your robot’s last position
- You want to save your heroes current hp
- You want to save you current level in you favourite console game

YOUR VARIABLES WILL DISAPPEAR AS SOON AS YOUR APPLICATION STOP RUNNING

Angga Raditya

LETS BACK TO YOUR HARDWARE





You want to save all those informations permanently in this f*cking box as aFILE

FILEOUTPUTSTREAM

```
public class Main {

    final static File fileCoba = new File("/Users/anggaraditya/coba.txt");


    public static void main(String[] args)  {

        try {
            FileOutputStream fos = new FileOutputStream(fileCoba);
            fos.write(65);
            fos.close();
        } catch (IOException e) {
            e.printStackTrace();
        }

    }

}
```

?

FILEOUTPUTSTREAM

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

FILEOUTPUTSTREAM

- It the basic writer in Java IO
- It can write only a byte or a char with an ascii code

FILEWRITER

```
public class Main {

    final static File fileCoba = new File("/Users/anggaraditya/coba.txt");

    public static void main(String[] args)  {
        try {
            FileWriter fw = new FileWriter(fileCoba);
            fw.write("Angga Raditya");
            fw.close();
        } catch (IOException e) {
            e.printStackTrace();
        }

    }

}
```

FILEWRITER

- It use FileOutputStream
- It gives you few feature sa: write(String);
- But still, it only append(Char)

Hints : find out write() and append() usage

BUFFEREDWRITER

```
public class Main {

    final static File fileCoba = new File("/Users/anggaraditya/coba.txt");

    public static void main(String[] args)  {
        try {
            BufferedWriter fw = new BufferedWriter(new FileWriter(fileCoba,true));
            fw.newLine();
            fw.append("Seorang penikmat senja");
            fw.close();
        } catch (IOException e) {
            e.printStackTrace();
        }

    }

}
```

BUFFEREDWRITER

- It use FileWriter
- I guess you can find out by yourself

THE READER

FileOutputStream

↓

FileWriter

↓

BufferedWriter

FileInputStream

↓

FileReader

↓

BufferedReader

Penulis

Pembaca

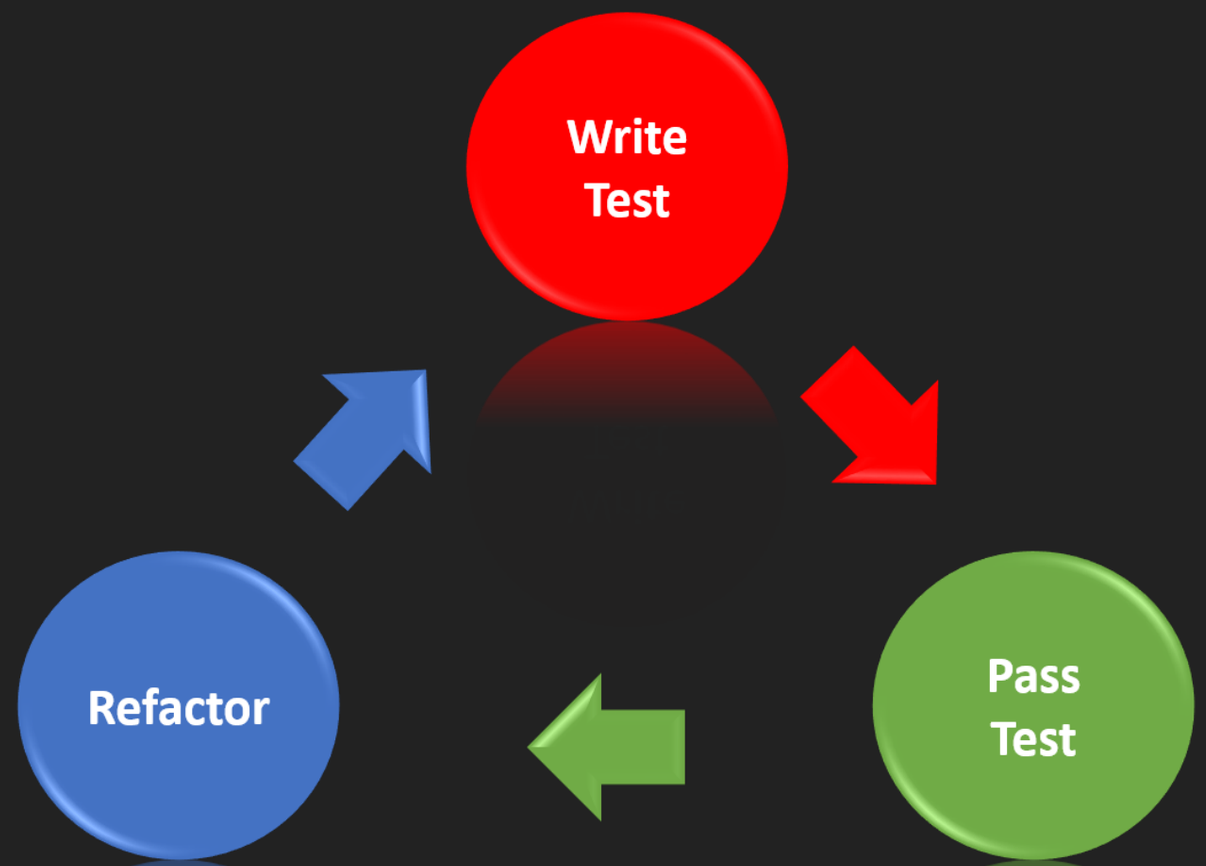
```java
public void read()  {
    try {
        FileInputStream fileInputStream = new FileInputStream(fileCoba);

        while (true){
            int ascii = fileInputStream.read();
            if(ascii==-1) break;
            System.out.println( (char) ascii );
        }


    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

}
```

```java
public void read()  {
    try {
        FileReader fileReader = new FileReader(fileCoba);

        while (true){
            int ascii = fileReader.read();
            if(ascii==-1) break;
            System.out.println( (char) ascii );
        }


    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

}
```

```java
public void read()  {
    try {
        BufferedReader bufferedReader = new BufferedReader(new FileReader(fileCoba));

        while (true){
            String text = bufferedReader.readLine();
            if(text==null) break;
            System.out.println(text);
        }


    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }

}
```

TEST DRIVEN DEVELOPMENT (TDD)

WHAT IS'T TDD ??



THE CYCLES OF TDD :

TDD is a very simple but powerful technique. It consists of 3 steps:

- Write a **minimal** test, which fails (red)
- Write the **minimal** code to fulfill the test (green)
- Refactor (green)

“ I CAN'T IMAGE WRITING CODE WITHOUT HAVING A TEST FIRST ANYMORE. ”

1. MINIMAL TEST

If you want to know the available space in parking area.

2. MINIMAL CODE

Sticking to our above example of to the available space in parking area, then you must count the number of parking slots in the area when there are no cars parked.

3. REFACTOR

NEVER CHANGE BOTH AT THE SAME TIME. AND THE ONLY OTHER CONDITION IS: DO NOT BREAK ANYTHING.

JUnit …

JUnit is a test framework which uses annotations to identify methods that specify a test in java.

How to Define a test in JUnit ??

A JUnit test is a method contained in a class which only used for the testing. This is called a Test *class*. To define that a certain method is a test method, annotate it with @TEST annotation.

What is Junit Assert?

Assert is a method useful in determining Pass or Fail status of a test case.

Assert Equals

If you want to test equality of two objects, you have the following methods

assertEquals(expected, actual)

It will return true if: **expected.equals(actual)** returns true.

EXAMPLE JUnit TEST

```java
import static org.junit.Assert.assertEquals;

public class ParkingAreaTest {

    @Test
    public void countAvailable_return_1_when_noCarPark() {
        Integer maximumSlot = 1;
        ParkingArea parkingArea = new ParkingArea(maximumSlot);
        assertEquals(maximumSlot, parkingArea.countAvailableSpace());
    }
```

What ? This is an error guys -_-

The solution is ..

```java
public class ParkingAreaTest {

    @Test
    public void countAvailable_return_1_when_noCarPark() {
        Integer maximumSlot = 1;
        ParkingArea parkingArea = new ParkingArea(maximumSlot);
        assertEquals(maximumSlot, parkingArea.countAvailableSpace());
    }
```

Click, and create a method.

```java
public class ParkingArea {
    private Integer maximumSlot;
    private Set<Car> carSlots = new HashSet<>();

    public ParkingArea(Integer maximumSlot) { this.maximumSlot = maximumSlot; }

    public Integer countAvailableSpace() {
        return maximumSlot - carSlots.size();
    }
```

For check the test case :

```java
@Test
public void countAvailable_return_1_when_noCarPark() {
    Integer maximumSlot = 1;
    ParkingArea parkingArea = new ParkingArea(maximumSlot);
    assertEquals(maximumSlot, parkingArea.countAvailableSpace());
}
```

```
✔ Tests passed: 1 of 1 test – 2 ms
"C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2019.2.2\jbr\bin\java.exe" ...



Process finished with exit code 0
```

7 Popular Unit Test Naming Conventions

1. MethodName_StateUnderTest_ExpectedBehavior
2. MethodName_ExpectedBehavior_StateUnderTest
3. test[Feature being tested]
4. Feature to be tested
5. Should_ExpectedBehavior_When_StateUnderTest
6. When_StateUnderTest_Expect_ExpectedBehavior
7. Given_Preconditions_When_StateUnderTest_Then_Expected Behavior: