Q) How are classes different from objects in Java? Explain with the help of real world example.

Classes vs Objects

Class

- Logical Entity
- do not occupy stack or heap memory space
- Blueprint for an Object
  - Data Members (Properties)
  - Member functions (Behavior)

Object

- Physical / Real-world entity
- Occupies both stack & heap memory space
- Instance of a class
  - Reference Variable in Stack
  - Actual Object in Heap

```java
class Movie{
    int duration;
    String name;
    double ratings;
    String genre;
}
```

Class { code → Text Segment }
(Class Loading)

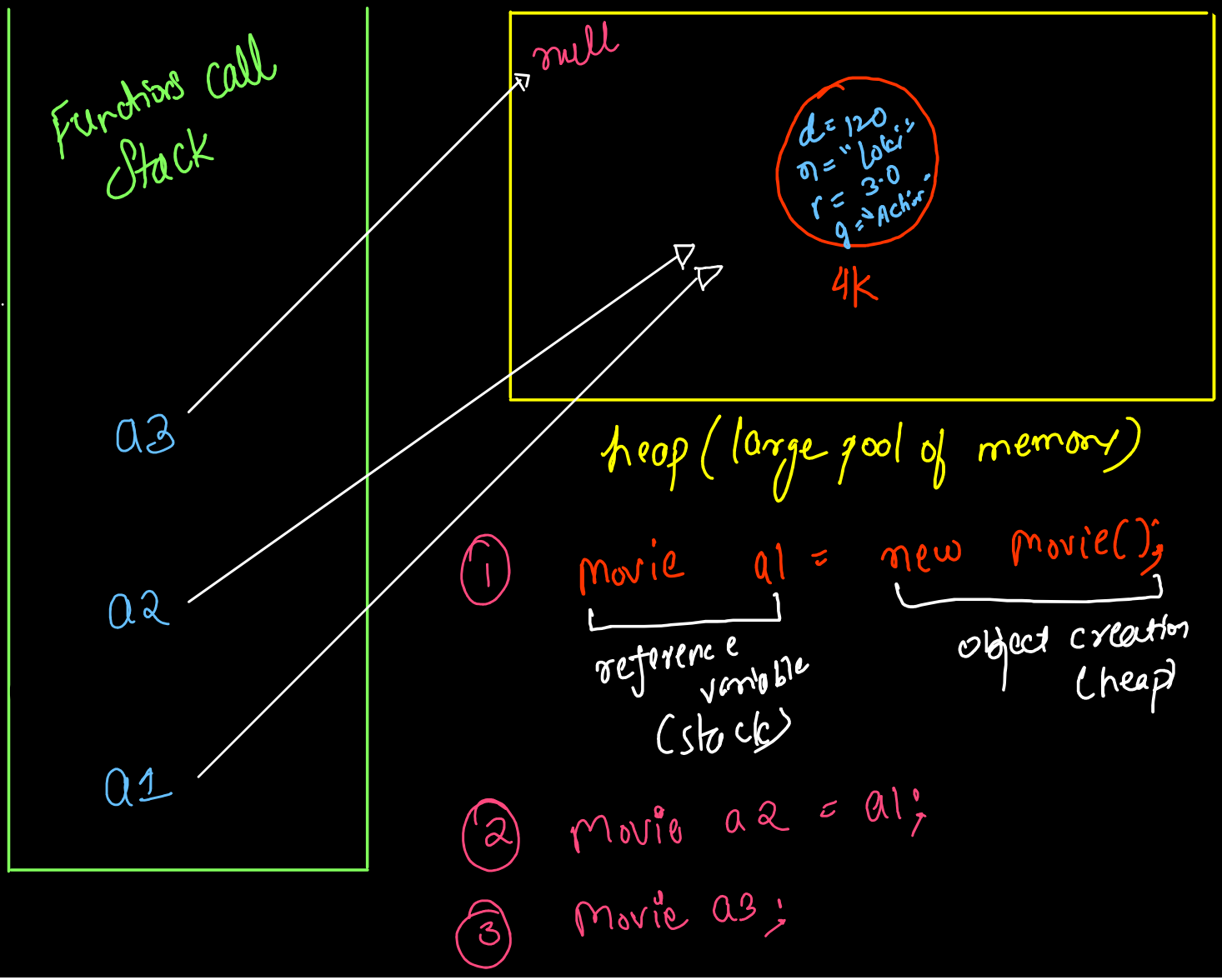
```java
Movie avengers1 = new Movie();
avengers1.duration = 120;
avengers1.name = "Avengers Loki";
avengers1.ratings = 4.0;
avengers1.genre = "Action";

// System.out.println(avengers1);
System.out.println(avengers1.name + " " +
                   avengers1.duration + " " +
                   avengers1.genre + " " +
                   avengers1.ratings);
```

Object

Memory Allocation

{ Reference Variable → Stack
Actual Instance → Heap }

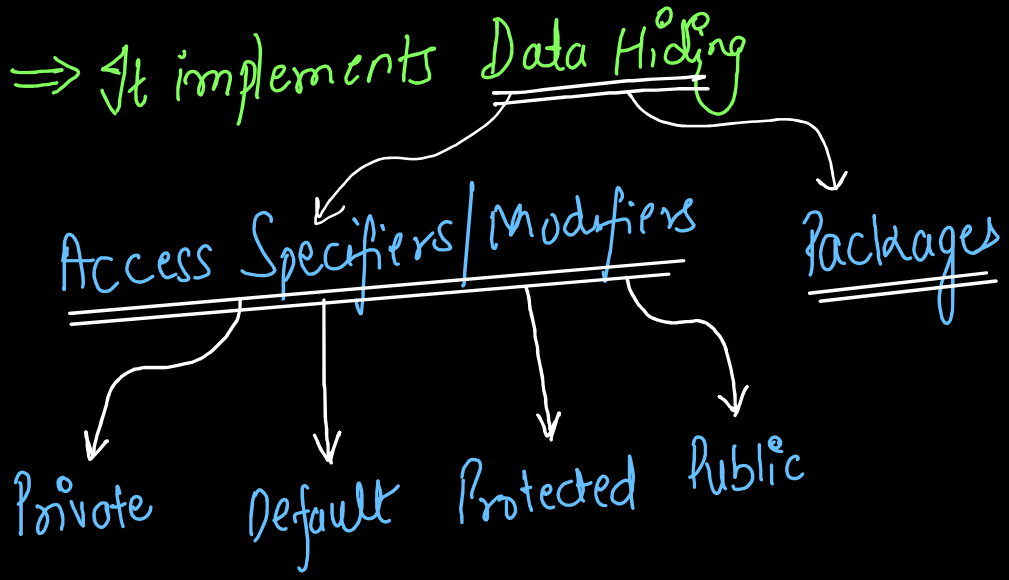
```
Avengers Loki 120 Action 4.0
```

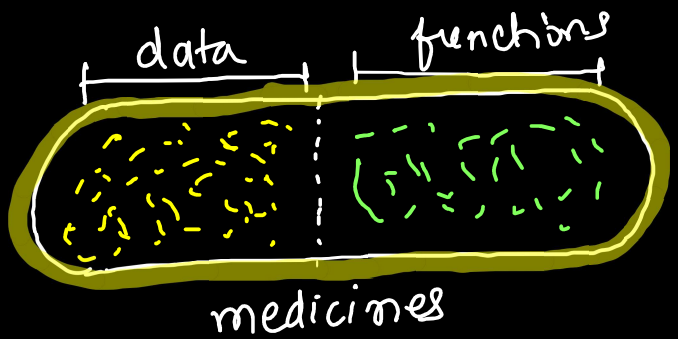

Function call Stack
a3
a2
a1
null
d = 120
n = "Loki"
r = 3.0
g = "Action"
4k
heap (large pool of memory)
① Movie a1 = new Movie();
reference variable (stack)
object creation (heap)
② Movie a2 = a1;
③ Movie a3;

Q) What is ENCAPSULATION? Explain it with the help of real world example!

"Wrapping together the data members and member functions into a single entity (known as class)"

⇒ It implements Data Hiding





```java
public static void main(String[] args) {
    int[] durations = {120, 150, 140, 200, 130};

    String[] name = {"Avengers Loki",
                     "Avengers Ultron",
                     "Avengers Infinity War",
                     "Avengers Endgame",
                     "Thor Love and Thunder"};

    double[] ratings = {4.0, 3.0, 4.9, 5.0, 2.0};

    String[] genre = {"Action", "Comedy", "Thriller", "Scifi", "Romance"};

    System.out.println(name[4] + " " +
                       durations[4] + " " +
                       genre[4] + " " +
                       ratings[4]);
}
```

```
Finished in 136 ms
Thor Love and Thunder 130 Romance 2.0
```

Code Without Encapsulation (Without Classes & Objects)

```java
class Movie{
    private int duration;
    private String name;
    private double rating;
    private String genre;

    public void setDuration(int newDuration){ duration = newDuration; }
    public void setName(String newName){ name = newName; }
    public void setRating(double newRating){ rating = newRating; }
    public void setGenre(String newGenre){ genre = newGenre; }

    public int getDuration(){ return duration; }
    public String getName(){ return name; }
    public double getRating(){ return rating; }
    public String getGenre(){ return genre; }
}
```

Data members (Properties) ⇒ Private

setters

getters

Member Functions (Behavior) (manipulators) ⇓ Public

```java
Movie avengers1 = new Movie();
avengers1.setDuration(120);
avengers1.setName("Avengers Loki");
avengers1.setRating(4.0);
avengers1.setGenre("Action");

System.out.println(avengers1.getName() + " "
                   + avengers1.getDuration() + " " +
                   avengers1.getGenre() + " " +
                   avengers1.getRating());
```

Q (A) What are CONSTRUCTORS in Java? Explain with a coding example.

→ Special function → same name as the classname

→ Provide default values to the properties.

→ Called as soon as Object is Created

→ for an Object, constructor is only called once

→ no "explicit" return type (But there is an implicit return type)

→ It is a special function which gets called as soon as the object is allocated heap memory.

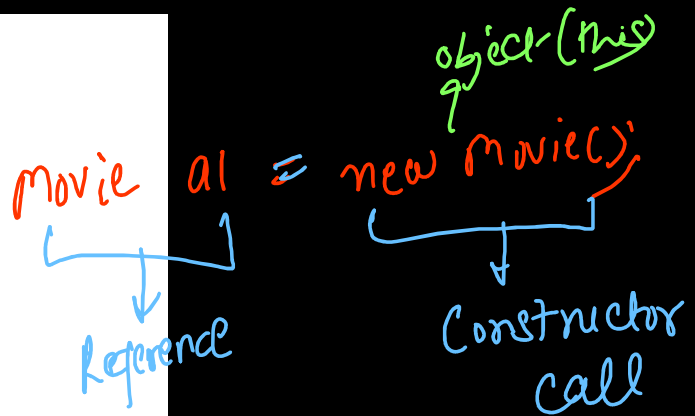
→ It has same name as classname.

→ It has no explicit return type.

→ Implicitly, return type of constructor is "this" (reference of the class current object)

→ Constructor cannot be static/abstract/final.
However, it can be private/protected/default.

```java
class Movie{
    public Movie(){
        duration = 100;
        name = "Untitled";
        rating = 0.0;
        genre = "Unclassified";
    }

    private int duration;
    private String name;
    private double rating;
    private String genre;

    public void setDuration(int newDuration){ duration = newDuration; }
    public void setName(String newName){ name = newName; }
    public void setRating(double newRating){ rating = newRating; }
    public void setGenre(String newGenre){ genre = newGenre; }

    public int getDuration(){ return duration; }
    public String getName(){ return name; }
    public double getRating(){ return rating; }
    public String getGenre(){ return genre; }
```



Q(B) What are the types of constructors in java? Write implementation of all of them.

Types of Constructors

- Default Implicit Constructor → no parameter, no body
- Default Explicit Constructor → no parameters
- Parameterized Constructor → 1 or more than 1 parameters
- Copy constructor → Parameter of reference of same class' object

(A) Default Implicit Constructor

```
public Movie(){}
```

(B) Default Explicit Constructor

```
public Movie(){
    duration = 100;
    name = "Untitled";
    rating = 0.0;
    genre = "Unclassified";
}
```

(C) Parameterized Constructor

```
public Movie(int duration, String name,
             double rating, String genre){

    setDuration(duration);
    setName(name);
    setRating(rating);
    setGenre(genre);
}
```

(D) Copy Constructor

```
public Movie(Movie other){
   setDuration(other.getDuration());
   setName(other.getName());
   setRating(other.getRating());
   setGenre(other.getGenre());
}
```

Q) What is constructor overloading? What are the rules for overloading of two methods/constructors?

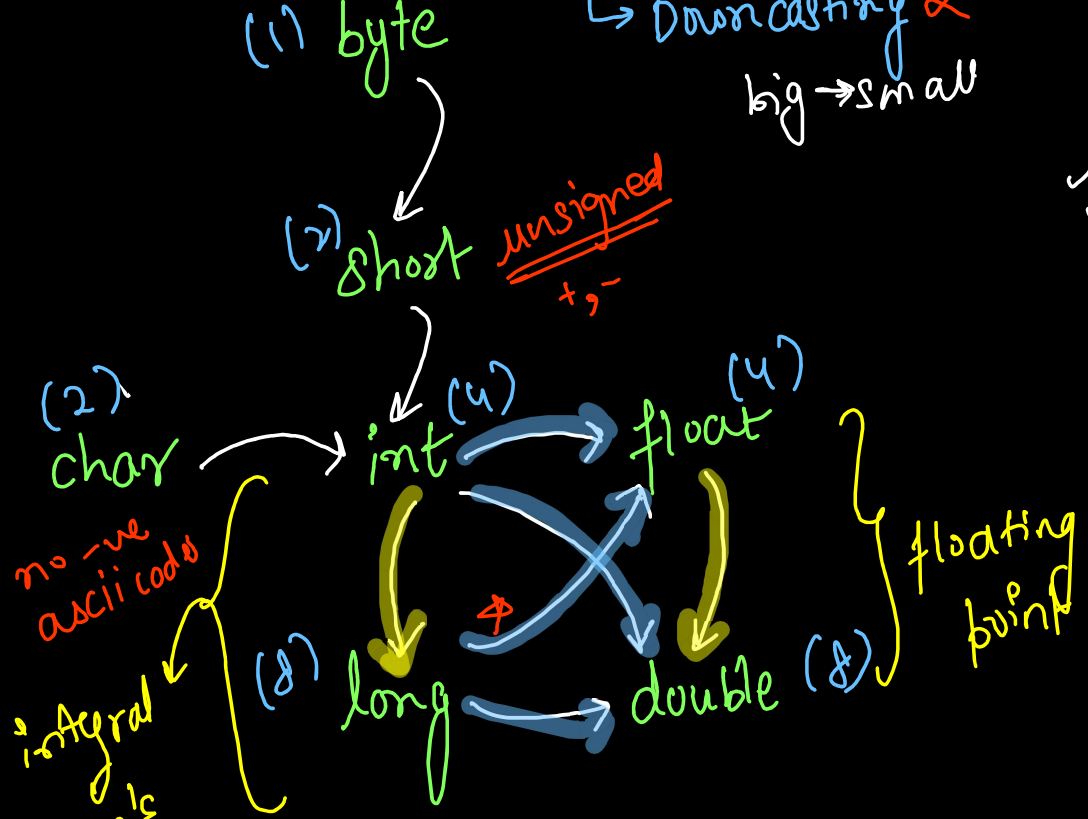
★ → Function name should be same

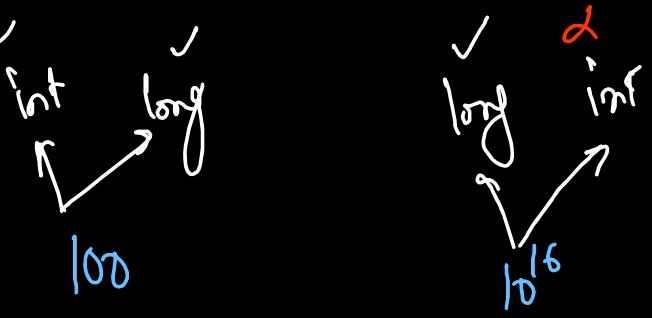
- Two constructors are said to be overloaded if atleast one condition is satisfied :→
  - → Number of arguments are different
  - → Types of arguments are different
  - → Order of arguments are different

★ → Change in return type "does not" make functions/methods overloaded!

Type Promotion in Java ⇒ to resolve ambiguity and match function call to a particular function definition

- Upcasting ✓ small → big
- Downcasting ✗ big → small





int
52
5200

→ float
52.0
5200.0

To put it another way, the JLS distinguishes between a loss of ***magnitude*** and a loss of ***precision***.

`int` to `byte` for example is a (potential) loss of magnitude because you can't store 500 in a `byte`.

`long` to `float` is a potential loss of precision but not magnitude because the value range for floats is larger than that for longs.

So the rule is:

- **Loss of magnitude:** explicit cast required;
- **Loss of precision:** no cast required.

```java
class Movie {
    int duration;
    String name, genre;
    double rating;

    public Movie(int duration) {
        this.duration = duration;
    }

}
```

```java
class Driver {
    Run | Debug
    public static void main(String[] args) {
        // NO EXACT MATCH FOUND: Movie(char)
        // Char Type Promoted to Integer (Upcasting - IMPLICIT)

        Movie avengers1 = new Movie(duration: 'A');
        System.out.println(avengers1.duration);

        // COMPILATION ERROR: Long Demoted to Integer
        // (Downcasting - IMPLICITLY NOT POSSIBLE)
        // Movie avengers2 = new Movie(180l);
        // System.out.println(avengers2.duration);

        // NO EXACT MATCH FOUND: Movie(long)
        // Long Type Demoted to Integer (Downcasting - EXPLICIT)

        Movie avengers2 = new Movie((int) 180l);
        System.out.println(avengers2.duration);
    }
}
```

```
architaggarwal@Archits-MacBook-Air Java OOPS % javac OOPS_Codes/5.TypePromotion.java
architaggarwal@Archits-MacBook-Air Java OOPS % java OOPS_Codes.Driver
65
180
```

Q) (A) Give the corrected output for the following code out of the given options.

Code :→

```
public static void swap(Movie a1, Movie a2){
    Movie a3 = a1;
    a1 = a2;
    a2 = a3;
}
```

```
Movie a1 = new Movie();
a1.setDuration(120);
System.out.println(a1.getDuration());

Movie a2 = new Movie();
a2.setDuration(150);
System.out.println(a2.getDuration());

swap(a1, a2);

System.out.println(a1.getDuration());
System.out.println(a2.getDuration());
```

options :→

(a) 120, 150, 150, 120

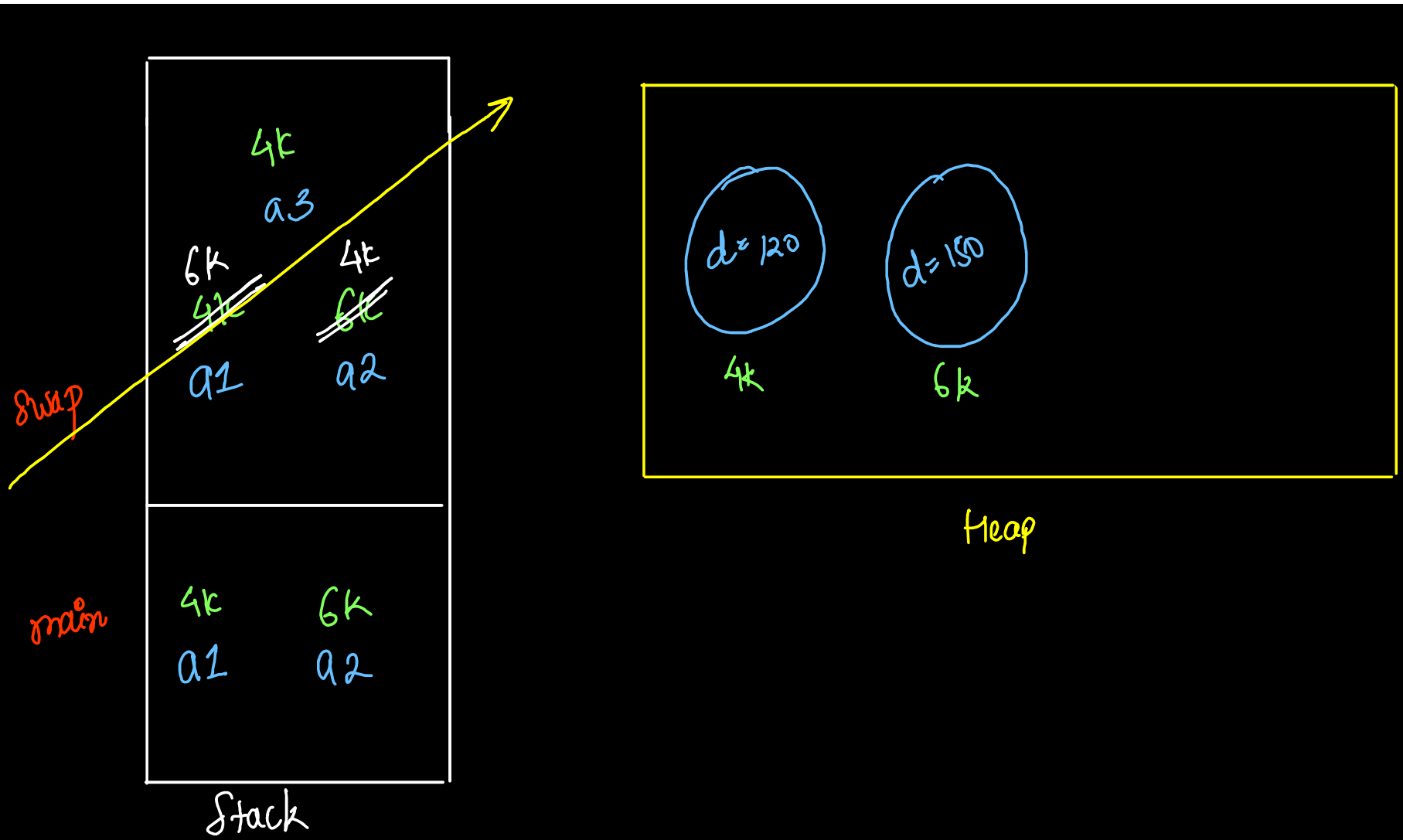
(b) 120, 150, 120, 150

(c) 120, 150, 120, 120

(d) 120, 150, 150, 150

Java is always pass by value!
↓
Stack changes does not persist!

Swap Game- (1)


4k
a3
6k
4k
4k
6k
a1
a2
swap
main
4k
6k
a1
a2
Stack
d = 120
d = 150
4k
6k
Heap

Q) (3) Give the corrected output for the following code out of the given options.

Code :→

```
public static void swap(Movie a1, Movie a2){
    Movie a3 = new Movie();
    a3.setDuration(a1.getDuration());

    a1 = a2;
    a2 = a3;
}
```

```
Movie a1 = new Movie();
a1.setDuration(120);
System.out.println(a1.getDuration());

Movie a2 = new Movie();
a2.setDuration(150);
System.out.println(a2.getDuration());

swap(a1, a2);

System.out.println(a1.getDuration());
System.out.println(a2.getDuration());
```

options :→

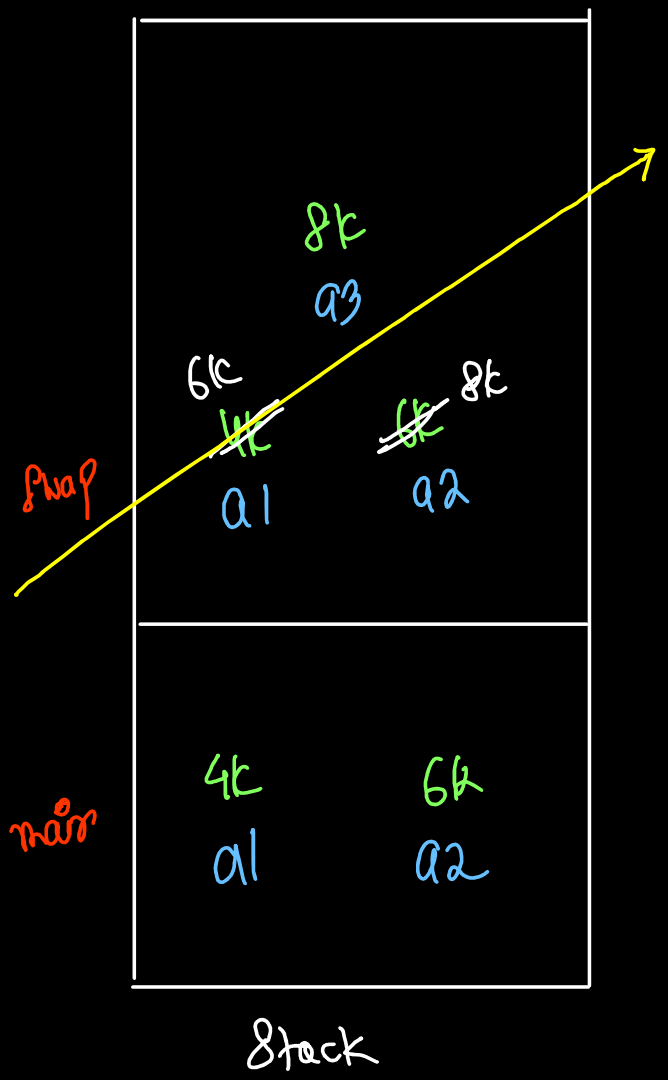
(a) 120, 150, 150, 120

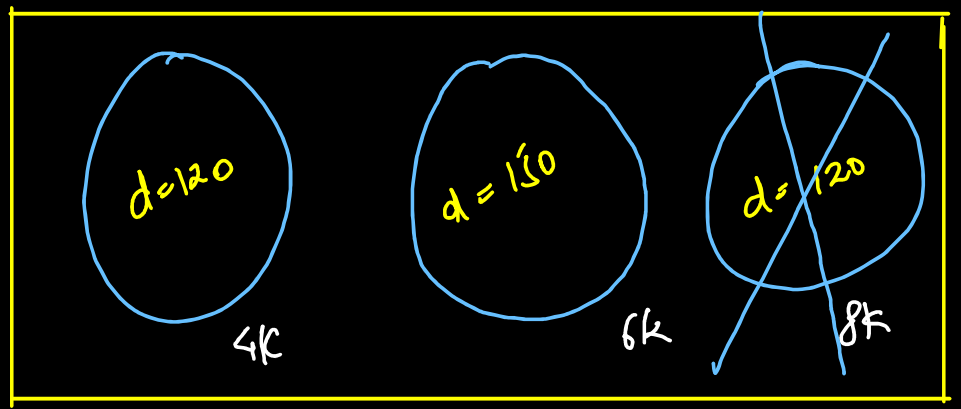
(b) 120, 150, 120, 150

(c) 120, 150, 120, 120

(d) 120, 150, 150, 150

Swap Game - (2)





Heap

Object Creation
↳ swap
↳ garbage collectible

Q) Give the corrected output for the following code out of the given options.

Code :→

```
public static void swap(Movie a1, Movie a2){
    Movie a3 = a1;

    a1.setDuration(a2.getDuration());
    a2.setDuration(a3.getDuration());
}
```

```
Movie a1 = new Movie();
a1.setDuration(120);
System.out.println(a1.getDuration());

Movie a2 = new Movie();
a2.setDuration(150);
System.out.println(a2.getDuration());

swap(a1, a2);

System.out.println(a1.getDuration());
System.out.println(a2.getDuration());
```

options :→

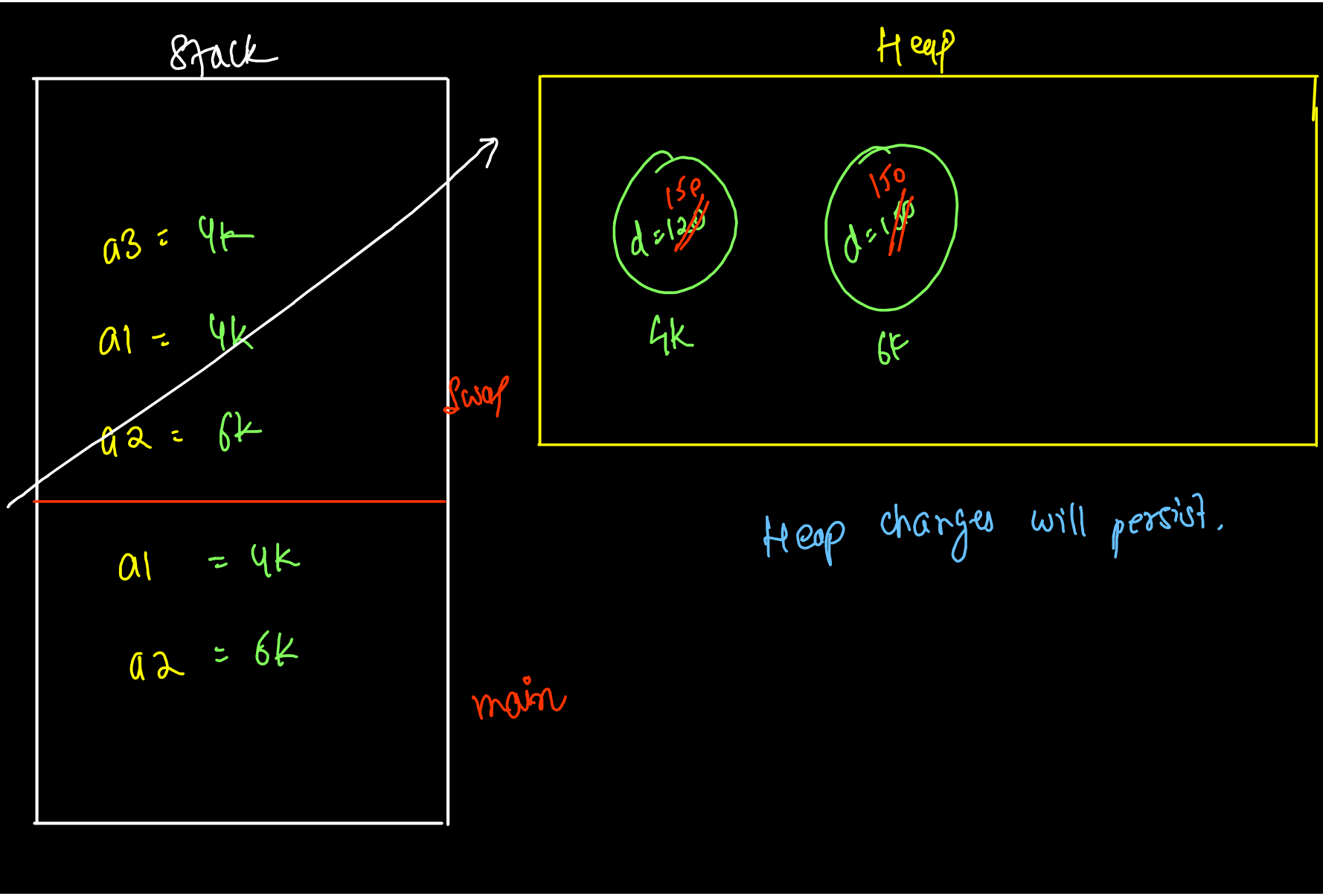
(a) 120, 150, 150, 120

(b) 120, 150, 120, 150

(c) 120, 150, 120, 120

(d) 120, 150, 150, 150

Swap Game → (3)


Stack
Heap
a3 = 4k
a1 = 4k
a2 = 6k
Swap
d = 150
d = 150
4k
6k
a1 = 4k
a2 = 6k
main
Heap changes will persist.

Q) ⓓ Give the corrected output for the following code out of the given options.

Code :→

```
public static void swap(Movie a1, Movie a2){
    Movie a3 = new Movie();
    a3.setDuration(a1.getDuration());

    a1.setDuration(a2.getDuration());
    a2.setDuration(a3.getDuration());
}
```

```
Movie a1 = new Movie();
a1.setDuration(120);
System.out.println(a1.getDuration());

Movie a2 = new Movie();
a2.setDuration(150);
System.out.println(a2.getDuration());

swap(a1, a2);

System.out.println(a1.getDuration());
System.out.println(a2.getDuration());
```

options :→

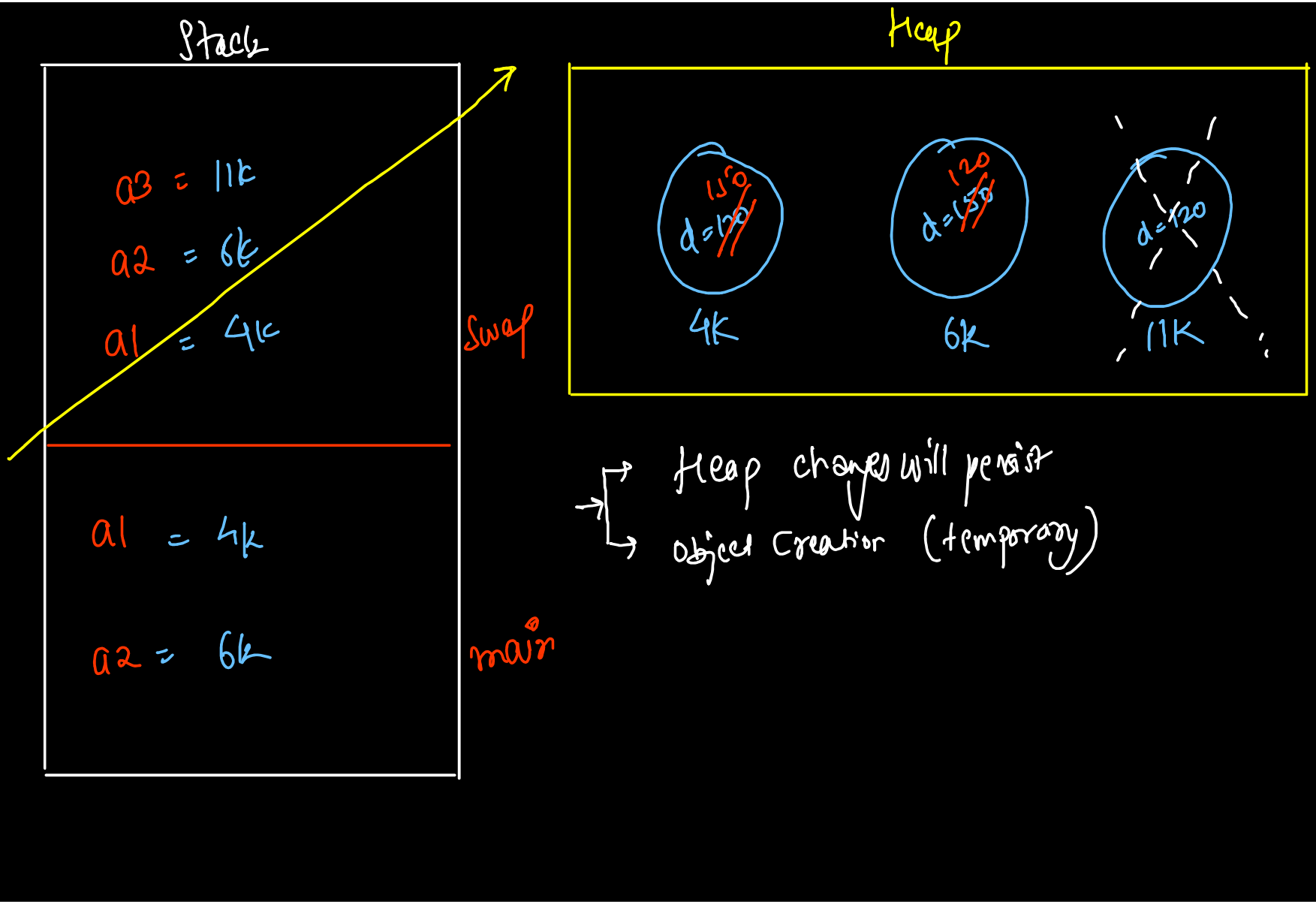
~~(a) 120, 150, 150, 120~~

(b) 120, 150, 120, 150

(c) 120, 150, 120, 120

(d) 120, 150, 150, 150

Swap Games - (4)

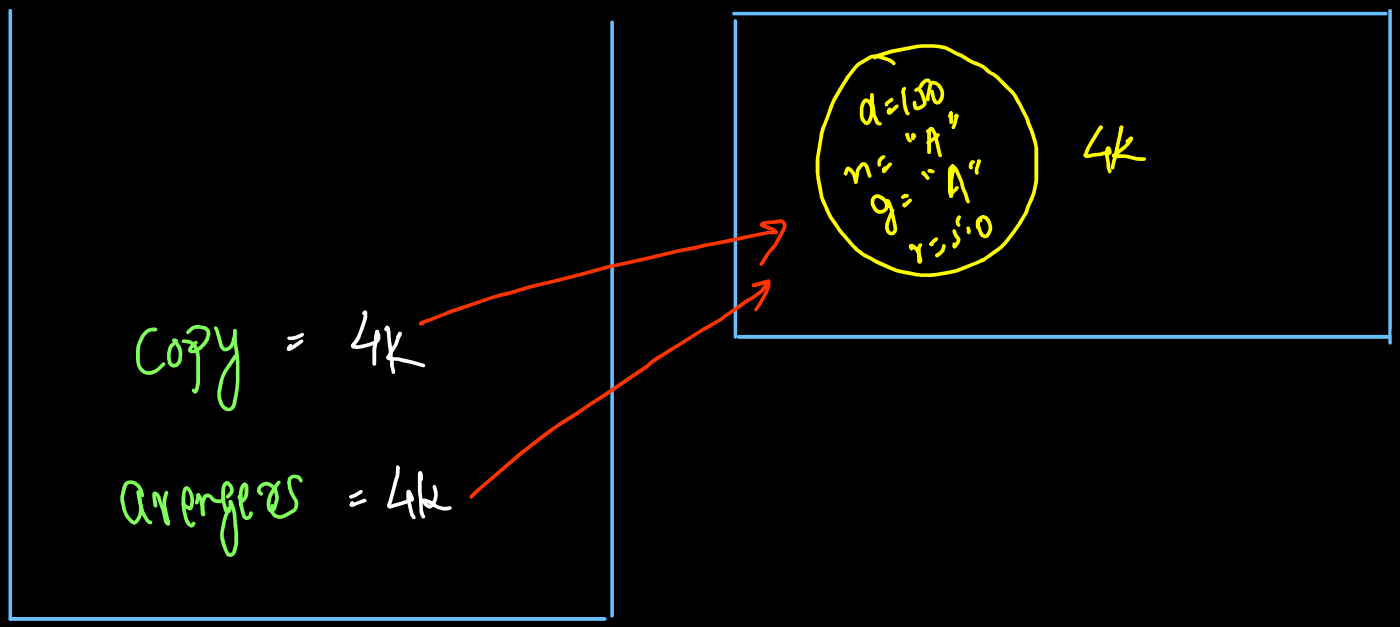

Stack
a3 = 11k
a2 = 6k
a1 = 4k
swap
a1 = 4k
a2 = 6k
main
Heap
150
d=120
4K
120
d=150
6K
d=120
11K
Heap changes will persist
Object Creation (temporary)

Q) What is the difference between Shallow copy and deep copy ?

Shallow Copy :-

- It stores the references of object to the original memory address.
- Changes made in the cloned new object are reflected in the original old object and vice-versa.
- It is just copy of reference variables, and not actual object creation is taking place
- Shallow copy is faster than deep copy!

```java
Movie avengers = new Movie();
System.out.println(avengers.duration + " " + avengers.genre
                    + " " + avengers.name + " " + avengers.ratings);

Movie copy = avengers;
System.out.println(copy.duration + " " + copy.genre
                    + " " + copy.name + " " + copy.ratings);
```



```java
class Movie{
    int duration = 150;
    String genre = "Action", name = "Avengers";
    double ratings = 5.0;
}

public class Main {
    public static void main(String[] args) {
        Movie avengers = new Movie();
        System.out.println(avengers.duration + " " + avengers.genre
                           + " " + avengers.name + " " + avengers.ratings);

        Movie copy = avengers;
        System.out.println(copy.duration + " " + copy.genre
                           + " " + copy.name + " " + copy.ratings);

        copy.duration = 180;
        System.out.println(avengers.duration + " " + avengers.genre
                           + " " + avengers.name + " " + avengers.ratings);
        System.out.println(copy.duration + " " + copy.genre
                           + " " + copy.name + " " + copy.ratings);
    }
```

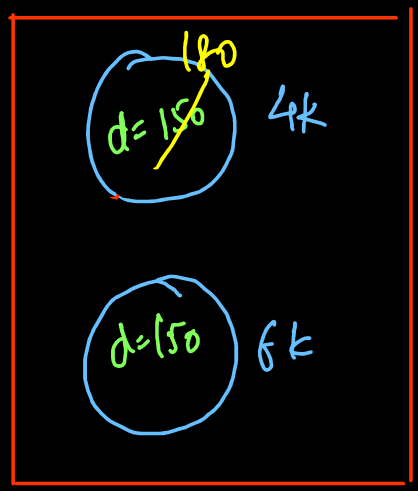
Deep Copy :-

- It creates a cloned object that is new data members (properties) & copies values in them.
- Changes in cloned new object are not reflected in changes of original object and vice-versa.
- It is actual copy of properties (stored in heap) and not just reference variables (in stack)
- Deep copy is slower than shallow copy.

```
    Movie(Movie other){
        duration = other.duration;
        genre = other.genre;
        name = other.name;
        ratings = other.ratings;
    }
}

public class Main {
    public static void main(String[] args) {
        Movie avengers = new Movie();
        System.out.println(avengers.duration + " " + avengers.genre
                           + " " + avengers.name + " " + avengers.ratings);

        Movie deepCopy = new Movie(avengers);
        System.out.println(deepCopy.duration + " " + deepCopy.genre
                           + " " + deepCopy.name + " " + deepCopy.ratings);

        avengers.duration = 180;
        System.out.println(avengers.duration + " " + avengers.genre
                           + " " + avengers.name + " " + avengers.ratings);
        System.out.println(deepCopy.duration + " " + deepCopy.genre
                           + " " + deepCopy.name + " " + deepCopy.ratings);
```

deepCopy = 6k

avengers = 4k



```java
class Movie {
    int duration;
    String name, genre;
    double rating;
    ArrayList<String> languages;

    public Movie(String name, int duration, double rating, String genre) {
        this.name = name;
        this.duration = duration;
        this.genre = genre;
        this.rating = rating;
        this.languages = new ArrayList<>();
    }

    // PARTIAL DEEP COPY CONSTRUCTOR
    public Movie(Movie other) {
        this.name = other.name;
        this.duration = other.duration;
        this.genre = other.genre;
        this.rating = other.rating;
        this.languages = other.languages;
    }
}
```

```
[English, Hindi]
architaggarwal@Archits-MacBook-Air Java OOPS % javac OOPS_Codes/6.1.ShallowCopy.java
architaggarwal@Archits-MacBook-Air Java OOPS % java OOPS_Codes.Driver
Avengers Endgame 180 4.5 SuperHero
[English]
[English, Tamil, Malayanam]
[English, Tamil, Malayanam]
Avengers Endgame 180 4.5 SuperHero
[English, Tamil, Malayanam]
[English, Tamil, Malayanam, Hindi, Telugu]
[English, Tamil, Malayanam, Hindi, Telugu]
architaggarwal@Archits-MacBook-Air Java OOPS %
```

```java
class Driver {
    Run | Debug
    public static void main(String[] args) {

        Movie avengers = new Movie(name: "Avengers Endgame", duration: 180,
                rating: 4.5, genre: "SuperHero");
        avengers.languages.add(e: "English");

        System.out.println(avengers.name + " " + avengers.duration
                + " " + avengers.rating + " " + avengers.genre);
        System.out.println(avengers.languages);

        // SHALLOW COPY

        Movie shallowCopy = avengers;

        avengers.languages.add(e: "Tamil");
        shallowCopy.languages.add(e: "Malayanam");

        System.out.println(shallowCopy.languages);
        System.out.println(avengers.languages);

        // PARTIAL DEEP COPY

        Movie partialDeep = new Movie(avengers);
        System.out.println(partialDeep.name + " " + partialDeep.duration
                + " " + partialDeep.rating + " " + partialDeep.genre);
        System.out.println(partialDeep.languages);

        avengers.languages.add(e: "Hindi");
        partialDeep.languages.add(e: "Telugu");

        System.out.println(partialDeep.languages);
        System.out.println(avengers.languages);
    }
}
```

Plain Old Java Object (POJO) class Features

Data Members & Member Functions in class

- Properties should be private!
- Getters & Setters should be public!

to implement "Data Hiding"

- Always create a default explicit constructor

- Function (vs) Method ~ Function inside classes associated with an object are known as methods!

Q) What are differences between methods & functions in Java ?

A method is a function or procedure in object oriented programming.

- Function inside a class which is called / invoked or associated with objects of that class is known as a method.
- Functions can be written inside as well as outside classes. ie. functions do not need a class to be created.
- Functions outside the class can only access local / primitive data like parameters whereas methods can access private data members of a class / object directly !

a) What is this keyword? What are it's applications?

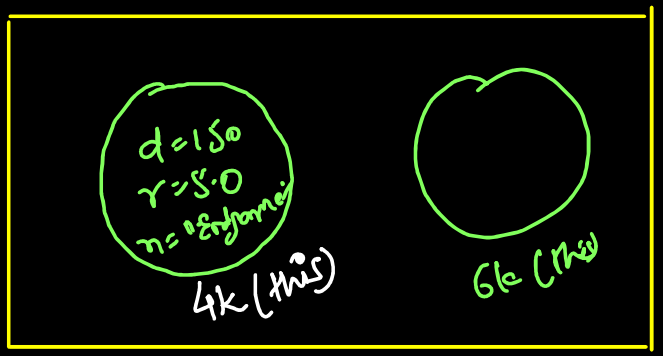
THIS Keyword ⇝ Self referential pointer (reference variable pointing to current object)

APPLICATIONS ⇒

1) → Refer data members from within the class → Resolve name conflict b/w parameters & data members
2) → Refer member functions from within the class
3) → Used in constructor chaining (invoke own Constructor) → chaining should be the first & single call only!
4) → Pass the current object as a parameter to a method/constructor
5) → Return the current object from a method/constructor → Return type of a constructor

```java
class Movie {
    int duration;
    double ratings;
    String name;
    String genre;

    public Movie() {
    }

    public Movie(int newDuration, double newRatings,
                 String newName, String newGenre) {
        duration = newDuration;
        ratings = newRatings;
        name = newName;
        genre = newGenre;
    }
}
```

IMPLICIT RETURN TYPE: THIS
(MOVIE REFERENCE TYPE)




```java
public static void main(String[] args) {
    Movie avengers = new Movie(newDuration: 150, newRatings: 5.0, newName: "Avengers Endgame",
    newGenre: "Thriller");
    System.out.println(avengers.duration + " " + avengers.name
            + " " + avengers.ratings + " " + avengers.genre);
}
```

```java
class Movie {
    private int duration;

    // Application 1: Constructor Return type
    // Application 2: Invoking Member Function
    public Movie(int duration) {
        this.setDuration(duration);
    }

    public int getDuration() {
        return duration;
    }

    // Application 3: Access Data Member Properties
    public void setDuration(int duration) {
        this.duration = duration;
    }

    // Application 4: Pass Current Object as Parameter
    public void display() {
        Driver.displayDurationOutside(this);
    }

    // Application 5: Return Current Object
    public Movie join(Movie other) {
        this.duration += other.duration;
        return this;
    }
}
```

```java
class Driver {
    public static void displayDurationOutside(Movie obj) {
        System.out.println("Movie Duration = " + obj.getDuration());
    }

    Run | Debug
    public static void main(String[] args) {
        Movie avengers1 = new Movie(duration: 120);
        avengers1.display();

        Movie avengers2 = new Movie(duration: 150);
        avengers1.join(avengers2);
        avengers1.display();
    }
}
```

```
architaggarwal@Archits-MacBook-Air System Design % javac OOPS_Codes/08.ThisKeyword.java
architaggarwal@Archits-MacBook-Air System Design % java OOPS_Codes.Driver
Movie Duration = 120
Movie Duration = 270
architaggarwal@Archits-MacBook-Air System Design %
```

Q) Are there default parameterized constructors or functions in Java? What do you understand by Constructor Chaining?

```java
class Cuboid{
    int length;
    int breadth;
    int height;

    Cuboid(){
        // this.length = 1;
        // this.breadth = 1;
        // this.height = 1;
        this(1);
    }

    Cuboid(int side){
        // this.length = side;
        // this.breadth = side;
        // this.height = side;
        this(side, side, side);
    }

    Cuboid(int length, int breadth){
        // this.length = length;
        // this.breadth = breadth;
        // this.height = 1;
        this(length, breadth, 1);
    }

    Cuboid(int length, int breadth, int height){
        this.length = length;
        this.breadth = breadth;
        this.height = height;
    }
}
```

Constructor Chaining

One constructor calling another constructor using this() keyword!

```java
public static void main(String[] args){
    Cuboid obj1 = new Cuboid();
    Cuboid obj2 = new Cuboid(5);
    Cuboid obj3 = new Cuboid(5, 10);
    Cuboid obj4 = new Cuboid(5, 10, 15);

    System.out.println(obj1.length + " " + obj1.breadth + " " + obj1.height);
    System.out.println(obj2.length + " " + obj2.breadth + " " + obj2.height);
    System.out.println(obj3.length + " " + obj3.breadth + " " + obj3.height);
    System.out.println(obj4.length + " " + obj4.breadth + " " + obj4.height);

}
```

```
Finished in 109 ms
1 1 1
5 5 5
5 10 1
5 10 15
```

⇒ Constructor chaining call should be the first statement in the calling constructor.

```java
class Cuboid {
    int length, breadth, height;

    Cuboid() {
        this.length = 1;
        this.breadth = 1;
        this.height = 1;
    }

    Cuboid(int side) {
        this.length = side;
        this.breadth = side;
        this.height = side;
    }

    Cuboid(int length, int breadth) {
        this.length = length;
        this.breadth = breadth;
        this.height = 1;
    }

    Cuboid(int length, int breadth, int height) {
        this.length = length;
        this.breadth = breadth;
        this.height = height;
    }
}
```

```java
public class Main {
    Run | Debug
    public static void main(String[] args) {
        Cuboid obj1 = new Cuboid();
        System.out.println(obj1.length + " " + obj1.breadth + " " + obj1.height);

        Cuboid obj2 = new Cuboid(side: 5);
        System.out.println(obj2.length + " " + obj2.breadth + " " + obj2.height);

        Cuboid obj3 = new Cuboid(length: 10, breadth: 15);
        System.out.println(obj3.length + " " + obj3.breadth + " " + obj3.height);

        Cuboid obj4 = new Cuboid(length: 10, breadth: 15, height: 20);
        System.out.println(obj4.length + " " + obj4.breadth + " " + obj4.height);

    }
}
```

Code without constructor chaining
Code redundancy!

```java
class Cuboid {
    int length, breadth, height;

    Cuboid() {
        this(side: 1);
    }

    Cuboid(int side) {
        this(side, side, side);
    }

    Cuboid(int length, int breadth) {
        this(length, breadth, height: 1);
    }

    Cuboid(int length, int breadth, int height) {
        this.length = length;
        this.breadth = breadth;
        this.height = height;
    }
}
```

```java
class Driver {
    Run | Debug
    public static void main(String[] args) {
        Cuboid obj1 = new Cuboid();
        Cuboid obj2 = new Cuboid(side: 5);
        Cuboid obj3 = new Cuboid(length: 5, breadth: 10);
        Cuboid obj4 = new Cuboid(length: 5, height: 10, breadth: 15);

        System.out.println(obj1.length + " " + obj1.breadth + " " + obj1.height);
        System.out.println(obj2.length + " " + obj2.breadth + " " + obj2.height);
        System.out.println(obj3.length + " " + obj3.breadth + " " + obj3.height);
        System.out.println(obj4.length + " " + obj4.breadth + " " + obj4.height);

    }
}
```

```
architaggarwal@Archits-MacBook-Air OOPS_Codes % javac 09.ConstructorChaining.java
architaggarwal@Archits-MacBook-Air OOPS_Codes % java Driver
1 1 1
5 5 5
5 1 10
5 15 10
```

Q) Is Java a PURE object oriented programming language? What are wrapper classes? What is autoboxing and unboxing?

Every entity should be class or object. And it should follow all object oriented programming principles!

int, boolean, char, long, float, double, short, byte

predefined (primitive) or userdefined

Answer: No

Solution ⇒ wrapper classes

int → Integer	float → Float
boolean → Boolean	double → Double
char → Character	short → Short
long → long	byte → Byte

Autoboxing & Unboxing

```java
public static void main(String[] args) {
    // Primitive Stack Variable
    int a = 5;
    System.out.print(a + " ");

    // Integer Wrapper Class
    Integer aa = 10; // Autoboxing
    System.out.print(aa + " ");

    a = aa; // Unboxing
    System.out.print(a + " ");

    // Actual Object Creation, Getter and Setter
    Integer b = new Integer(value: 20);
    System.out.print(b.toString() + " ");

    a = b.intValue();
    System.out.print(a + " ");
}
```

```
architaggarwal@Archits-MacBook-Air OOPS_Codes % javac 10.WrapperClasses.java
10.WrapperClasses.java:17: warning: [removal] Integer(int) in Integer has been deprecated and marked for removal
        Integer b = new Integer(20);
                    ^
1 warning
architaggarwal@Archits-MacBook-Air OOPS_Codes % java Driver

5 10 10 20 20 %
```

```java
public static void functionsAndConst() {
    Integer a = 10;

    // Integer to String and String to Integer
    String b = a.toString();
    System.out.println(a + " Integer to String : " + b);

    String c = "256";
    Integer d = Integer.parseInt(c);
    System.out.println(c + " in Integer : " + d);

    // Various Number Conversions
    System.out.println(a + " Decimal to Binary : " + Integer.toBinaryString(a));
    System.out.println(a + " Decimal to Hexadecimal : " + Integer.toHexString(a));
    System.out.println(a + " Decimal to Octal : " + Integer.toOctalString(a));

    System.out.println("Integer MAXIMUM Range : " + Integer.MAX_VALUE);
    System.out.println("Integer MINIMUM Range : " + Integer.MIN_VALUE);

    // Various Other Functions
    System.out.println(a.compareTo(d));
    System.out.println(a.equals(d));
    System.out.println(Integer.max(a, d));
    System.out.println(Integer.min(a, d));
}
```

```
architaggarwal@Archits-MacBook-Air OOPS_Codes % java Driver
10 Integer to String : 10
256 in Integer : 256
10 Decimal to Binary : 1010
10 Decimal to Hexadecimal : a
10 Decimal to Octal : 12
Integer MAXIMUM Range : 2147483647
Integer MINIMUM Range : -2147483648
-1
false
256
10
```

```java
class MyInteger {
    private int data;

    public MyInteger(int data) {
        this.data = data;
    }

    public int getData() {
        return data;
    }

    public void setData(int data) {
        this.data = data;
    }

}
```

```java
MyInteger val1 = new MyInteger(data: 50);
System.out.println(val1);
System.out.println(val1.getData());
val1.setData(data: 100);
System.out.println(val1.getData());
```

```
MyInteger@ea30797
50
100
```

```java
Integer val = new Integer(value: 5);
Integer val2 = 5; // autoboxing: Integer -> int
System.out.println(val2);

int val3 = val2; // Unboxing: int -> Integer
System.out.println(val3);
```

Q) What do you mean by static keyword? What are static variables or class variables in Java?

STATIC Keyword

→ Data members

→ Benefit :- memory efficiency

Property not associated with any object

Instance/object variable (nonstatic) vs Class variable (static)

Accessing static variables using classname (without object)

Eg Company name of Employees, college name of Students, language in Bollywood Movie Industry, etc.

```java
class Movie {
    // Non Static or Instance Variables
    int duration;
    String name;
    double rating;

    // Static or Class Variable
    static String language = "English";

    public Movie(String name, int duration, double rating) {
        this.name = name;
        this.duration = duration;
        this.rating = rating;
    }

}
```

```java
public static void main(String[] args) {
    Movie a1 = new Movie(name: "Avengers Infinity War", duration: 130, rating: 4.8);
    Movie a2 = new Movie(name: "Avengers End Game", duration: 200, rating: 4.9);
    Movie a3 = new Movie(name: "Avengers Secret Wars", duration: 170, rating: 4.6);
    Movie a4 = new Movie(name: "Avengers Kang Dynasty", duration: 220, rating: 5.0);

    // Non Static Variables
    System.out.println(a1.name + " " + a1.duration + " " + a1.rating);
    System.out.println(a2.name + " " + a2.duration + " " + a2.rating);
    System.out.println(a3.name + " " + a3.duration + " " + a3.rating);
    System.out.println(a4.name + " " + a4.duration + " " + a4.rating);

    // Static Variable or Class Variable
    System.out.print(a1.language + " ");
    System.out.print(a2.language + " ");
    System.out.print(a3.language + " ");
    System.out.print(a4.language + " ");

    // Accessing Using Class Name
    // (Static Properties Even Exist Without Single Object)
    System.out.print(Movie.language + " ");
}
```

```
architaggarwal@Archits-MacBook-Air OOPS_Codes % javac 11.StaticKeyword.java
architaggarwal@Archits-MacBook-Air OOPS_Codes % java Driver
Avengers Infinity War 130 4.8
Avengers End Game 200 4.9
Avengers Secret Wars 170 4.6
Avengers Kang Dynasty 220 5.0
English English English English English %
```

Interview Qbu

Q) Why are static variables considered evil?

- It represents global scope (accessible using classname & from all objects of class)
- lifetime of variable is very long (entire program)
- less Object-Oriented way and reduce reusability
  - class loading init
  - do not require object creation
  - not object specific
- Not thread safe and difficult to serialize
  (multiple threads will see same instance/copy of static variables which can lead to inconsistency/race condition during concurrent/parallel access).

```java
public static void staticEvil() {
    // 1. Program LifeTime
    // 2. Global Scope
    // 3. Less Object Oriented (Do Not Require Object Creation)
    System.out.println(Movie.language);

    Movie a1 = new Movie(name: "Avengers Infinity War", duration: 130, rating: 4.8);
    Movie a2 = new Movie(name: "Avengers End Game", duration: 200, rating: 4.9);
    Movie a3 = new Movie(name: "Avengers Secret Wars", duration: 170, rating: 4.6);
    Movie a4 = new Movie(name: "Avengers Kang Dynasty", duration: 220, rating: 5.0);

    System.out.println(a1.language + " " + a2.language + " " + a3.language + " " + a4.language);

    // 4. Same Copy Shared to Every Object
    a1.language = "Hindi";
    System.out.println(a1.language + " " + a2.language + " " + a3.language + " " + a4.language);

    // 5. Not Thread Safe and Non-Serializable
}
```

```
architaggarwal@Archits-MacBook-Air OOPS_Codes % javac 11.StaticKeyword.java
architaggarwal@Archits-MacBook-Air OOPS_Codes % java Driver
English
English English English English
Hindi Hindi Hindi Hindi
```

Q) What are static member functions in Java? What are rules for accessing static or non-static data?

Static Member functions

- Static method can access other static methods and static data members
- Non-static methods can access both non-static as well as static members (data members & other methods)
- Static methods can't access non-static methods & data members.
  - ↳ although we can still access by creating objects within static method.

Movie Class

```
public void nonStaticFun() {
    // Access Non Static Members (Properties & Other Functions) : Allowed
    System.out.println(" Name : " + this.name + " Duration : "
            + this.duration + " Rating : " + this.rating);

    // Access Static Member (Properties & Other Functions) : Allowed
    System.out.println("Language : " + Movie.language);
    Movie.staticFun();
}

public static void staticFun() {
    // This Keyword is Not There Inside Static Functions
    // System.out.println(this);

    // Access Non Static Members (Properties & Other Functions) : Not Allowed
    // System.out.println(" Name : " + this.name + " Duration : " + this.duration +
    // " Rating : " + this.rating);
    // nonStaticFun();

    // Access Static Member (Properties & Other Functions) : Allowed
    System.out.println("Language : " + Movie.language);

    // Access Non Static Members Using Object Creation
    Movie obj = new Movie(name: "Avengers", duration: 150, rating: 4.5);
    System.out.println(obj.name + " " + obj.duration + " " + obj.rating);
}
```

Output

```
architaggarwal@Archits-MacBook-Air OOPS_Codes % javac 11.StaticKeyword.java
architaggarwal@Archits-MacBook-Air OOPS_Codes % java Driver
Language : English
Avengers 150 4.5
 Name : Avengers Infinity War Duration : 130 Rating : 4.8
Language : English
Language : English
Avengers 150 4.5
```

Driver Code

```
// No Object Required for Calling Static Functions
Movie.staticFun();

Movie a1 = new Movie(name: "Avengers Infinity War", duration: 130, rating: 4.8);
// Object Required to Call Non Static Member Function
a1.nonStaticFun();
```

Non Static Methods

```
public void setLanguage(String newLanguage){ language = newLanguage; }
public String getLanguage(){ return language; }
```

(1) Using Objects → Variables & Funcn access : Allowed

```
System.out.println(avengers1.getLanguage());
avengers1.setLanguage("English");
System.out.println(avengers1.getLanguage());
```

(2) Classname (without object creation) → Not Allowed

```
System.out.println(Movie.getLanguage());
Movie.setLanguage("Tamil");
System.out.println(Movie.getLanguage());
```

You require objects/instances in order to access non static members

#STATIC METHODS

```
private static String language = "Hindi";
public static void setLanguage(String newLanguage){ language = newLanguage; }
public static String getLanguage(){ return language; }
```

1) Using Objects/Instances → Allowed ✓

```
System.out.println(avengers1.getLanguage());
avengers1.setLanguage("English");
System.out.println(avengers1.getLanguage());
```

2) Using Classname (without object creation) → Allowed ✓

```
System.out.println(Movie.getLanguage());
Movie.setLanguage("Tamil");
System.out.println(Movie.getLanguage());
```

You do not require object creation to access static members!

interview ques

Q) Why is main() method static in Java ?

entry point

- JVM can call it without object creation
- It is the first method to undergo execution
- JVM can call main() method directly using classname

Note :- It should also be always →
1. public
2. void return type
3. String [] args

"public static void main (String [] args)"

void → Default Exception Handler

C++
int main

0 → normal terminated

-1 or +1 → abnormal terminated

Interview Question

Q) Do we have "this" inside a "static" method?

A) No, since static methods are not associated with objects and this keyword is associated with a particular instance/object, hence we do not have "this" access inside "static" method.

Interview Question

Q) Can constructors be static?

A) No, since constructors implicitly return the current object reference ("this"), they cannot be static.

Q) What do you understand by static nested classes in Java?

Static
Nested class → To create objects (instantiate) inner class objects without instantiating outer class!

- Outer class can access inner class data members
- Inner class can only access outer class' static members
- Inner class object creation requires:- OuterclassName.innerclassName

Note: If main() method & inner class are in the same outer class, we can omit outerclass name

Outer class (Movie)

```java
class Movie {
    // Non Static or Instance Variables (Outer Class)
    int duration;
    String name;
    double rating;

    // Static or Class Variable (Outer Class)
    static String language = "English";

    public Movie(String name, int duration, double rating) {
        this.name = name;
        this.duration = duration;
        this.rating = rating;
    }

    public void outerNonStaticFun() {…

    public static void outerStaticFun() {…

    public static class Theater {…
}
```

```java
public void outerNonStaticFun() {
    // Access Outer Non Static Members (Properties & Other Functions) : Allowed
    System.out.println(" Name : " + this.name + " Duration : "
            + this.duration + " Rating : " + this.rating);

    // Access Outer Static Member (Properties & Other Functions) : Allowed
    System.out.println("Language : " + Movie.language);

    // Access Static Inner Class Static Members : Allowed
    System.out.println(Theater.screenType);

    // Access Static Inner Class Non Static Members
    // Without Inner Class Object: Not Allowed
    // System.out.println(this.chain + " " + this.noOfSeats);

    // With Inner Class Object: Allowed
    Theater inner = new Theater(chain: "PVR", noOfSeats: 100);
    System.out.println(inner.chain + " " + inner.noOfSeats);

    System.out.println(x: "------------------------");
}
```

```java
public static void outerStaticFun() {
    // Access Outer Non Static Members (Properties & Other Functions) : Not Allowed
    // System.out.println(" Name : " + this.name + " Duration : " + this.duration +
    // " Rating : " + this.rating);
    // outerNonStaticFun();

    // Access Outer Static Member (Properties & Other Functions) : Allowed
    System.out.println("Language : " + Movie.language);

    // Access Static Inner Class Static Members : Allowed
    System.out.println(Theater.screenType);

    // Access Static Inner Class Non Static Members
    // Without Inner Class Object: Not Allowed
    // System.out.println(this.chain + " " + this.noOfSeats);

    // With Inner Class Object: Allowed
    Theater inner = new Theater(chain: "PVR", noOfSeats: 100);
    System.out.println(inner.chain + " " + inner.noOfSeats);

    System.out.println(x: "------------------------");

}
```

Inner class (Theater)

```java
public static class Theater {
    // Non Static or Instance Variables (Inner Class)
    String chain;
    int noOfSeats;

    // Static or Class Variable (Inner Class)
    static String screenType = "2D";

    public Theater(String chain, int noOfSeats) {
        this.chain = chain;
        this.noOfSeats = noOfSeats;
    }

    public void innerNonStaticFun() {
        // Access Outer Non Static Members : Not Allowed
        // System.out.println(" Name : " + this.name + " Duration : "
        // + this.duration + " Rating : " + this.rating);

        // Access Outer Static Member (Properties & Other Functions) : Allowed
        System.out.println("Language : " + Movie.language);

        // Access Static Inner Class Static Members : Allowed
        System.out.println(Theater.screenType);

        // Access Static Inner Class Non Static Members: Allowed
        System.out.println(this.chain + " " + this.noOfSeats);

        System.out.println(x: "------------------------");
    }

    public static void innerStaticFun() {
        // Access Outer Non Static Members (Properties & Other Functions) : Not Allowed
        // System.out.println(" Name : " + this.name + " Duration : " + this.duration +
        // " Rating : " + this.rating);
        // outerNonStaticFun();

        // Access Outer Static Member (Properties & Other Functions) : Allowed
        System.out.println("Language : " + Movie.language);

        // Access Static Inner Class Static Members : Allowed
        System.out.println(Theater.screenType);

        // Access Static Inner Class Non Static Members
        // Without Inner Class Object: Not Allowed
        // System.out.println(this.chain + " " + this.noOfSeats);

        // With Inner Class Object: Allowed
        Theater inner = new Theater(chain: "PVR", noOfSeats: 100);
        System.out.println(inner.chain + " " + inner.noOfSeats);

        System.out.println(x: "------------------------");
    }
}
```

Driver code

```
class Driver {
    Run | Debug
    public static void main(String[] args) {
        Movie.outerStaticFun();
        Movie.Theater.innerStaticFun();

        Movie outer = new Movie(name: "Avengers", duration: 180, rating: 4.5);
        outer.outerNonStaticFun();

        Movie.Theater inner = new Movie.Theater(chain: "PVR", noOfSeats: 100);
        inner.innerNonStaticFun();
    }
}
```

Output

```
Language : English
2D
PVR 100
------------------------
Language : English
2D
PVR 100
------------------------
 Name : Avengers Duration : 180 Rating : 4.5
Language : English
2D
PVR 100
------------------------
Language : English
2D
PVR 100
------------------------
```

Q) What do you understand by "static block" in Java?

→ Static block
- used to initialize static variables
- executed before main() execution at time of class loading

Q) Can you run a java program only using static block, ie. without main function?

No, in newer Java versions, main() is mandatory to run a java program. If there is no entry point, java code will not run even if there is a static block.

```java
class Movie {
    // Non Static or Instance Variables
    int duration;
    String name;
    double rating;

    // Static or Class Variable
    static String language;

    public Movie(String name, int duration, double rating) {
        System.out.println(x: "Constructor - Object Creation");
        this.name = name;
        this.duration = duration;
        this.rating = rating;
        this.language = "Hindi";
    }

    // Static Book (Runs Before Main Methox Execution: During Loading & Linking)
    static {
        System.out.println(x: "Movie Static Block Executed");

        // Non Static Member Initialized
        language = "English";

        System.out.println(language + " OR " + Movie.language);
    }
}
```

```java
class Driver {
    static int x;
    static {
        System.out.println(x: "Driver Static block executed");
        x = 100;
        System.out.println(x);
    }

    Run | Debug
    public static void main(String[] args) {
        System.out.println(x: "Main Method Started");

        // main method must be there otherwise static block will also not run,
        // Although it will run after static block of it's class

        System.out.println(x);
        System.out.println(Movie.language);

        Movie obj = new Movie(name: "Avengers", duration: 180, rating: 4.5);
        System.out.println(obj.language);
    }
}
```

```
architaggarwal@Archits-MacBook-Air OOPS Codes % javac 11.3.StaticBlock.java
architaggarwal@Archits-MacBook-Air OOPS Codes % java Driver
Driver Static block executed
100
Main Method Started
100
Movie Static Block Executed
English OR English
English
Constructor - Object Creation
Hindi
```