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

Java Constructors

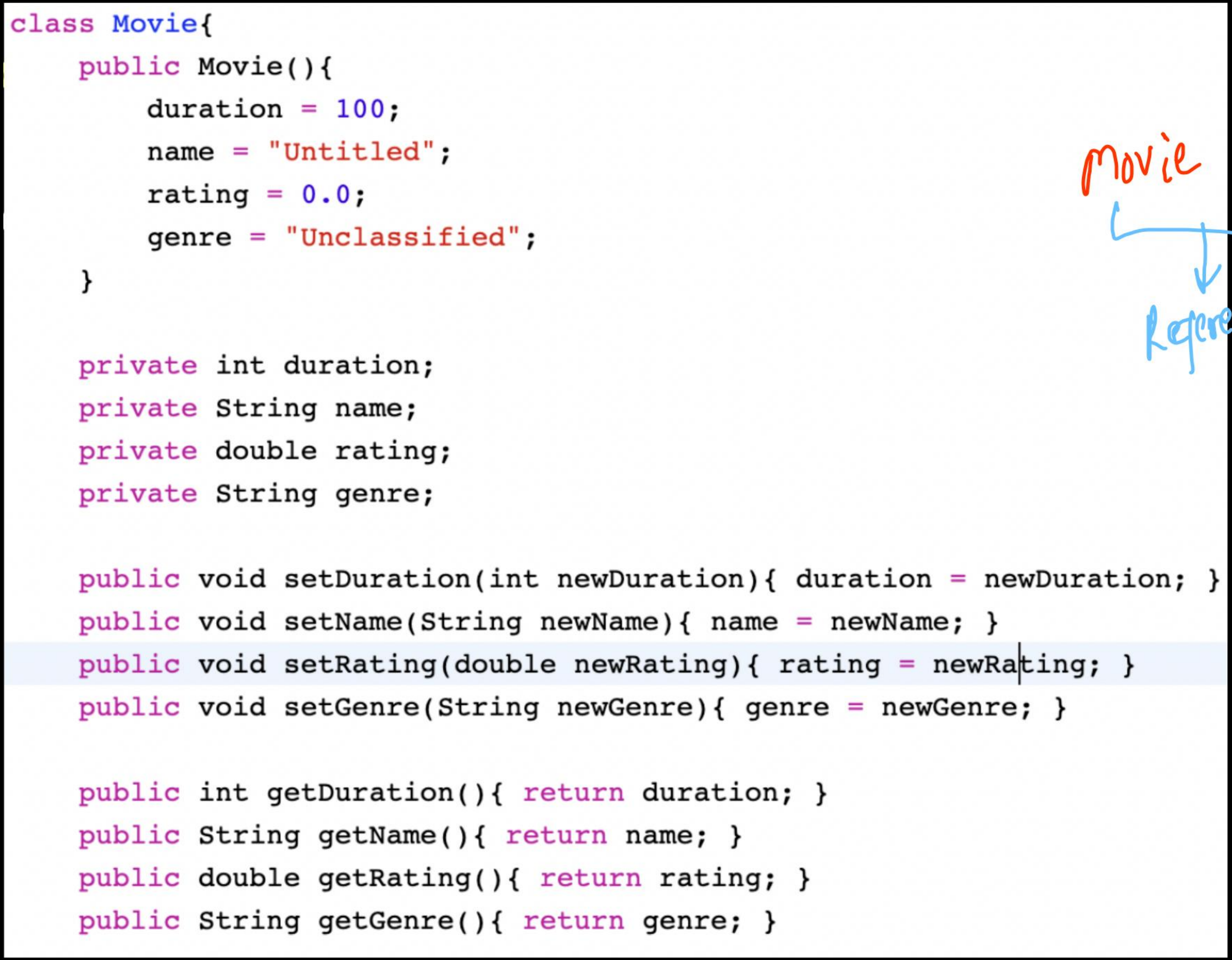
→ It is a special function which gets called as soon as the object is allocated heap memory.

→ It has same name as classname.

→ It has no explicit return type.

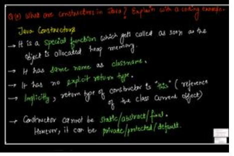
→ Implicitly, return type of constructor is "this" (reference of the class current object)

→ 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; }
```

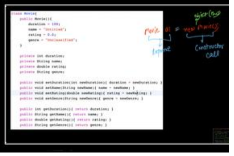
$$\underbrace{\text{Movie a1}}_{\text{Reference}} = \overset{\text{object (this)}}{\underbrace{\text{new Movie();}}_{\text{Constructor call}}}$$



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

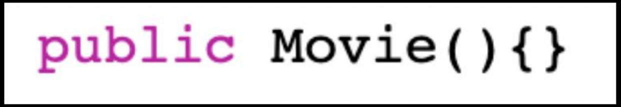
Types of Constructors

- 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



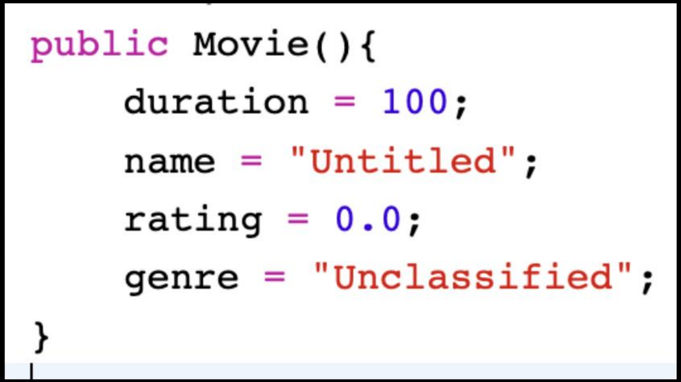
```
class Movie{
    public Movie(){
        duration = 100;
        name = "Untitled";
        rating = 0.0;
        genre = "Unclassified";
    }

    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; }
```

(A) Default Implicit Constructor



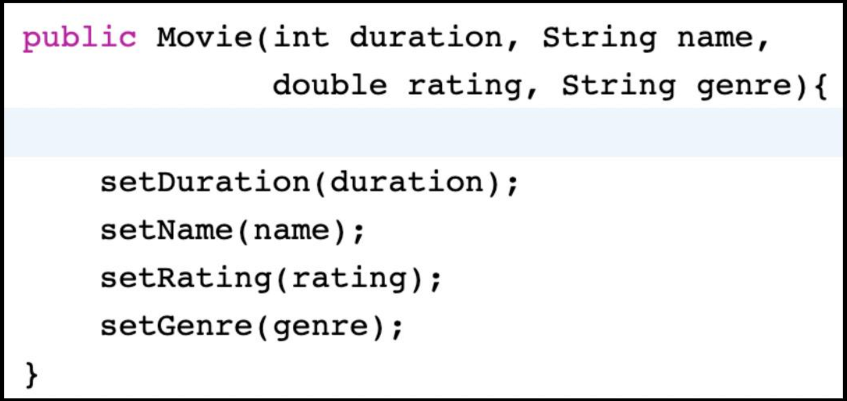
```java
public Movie(){}
```

(B) Default Explicit Constructor



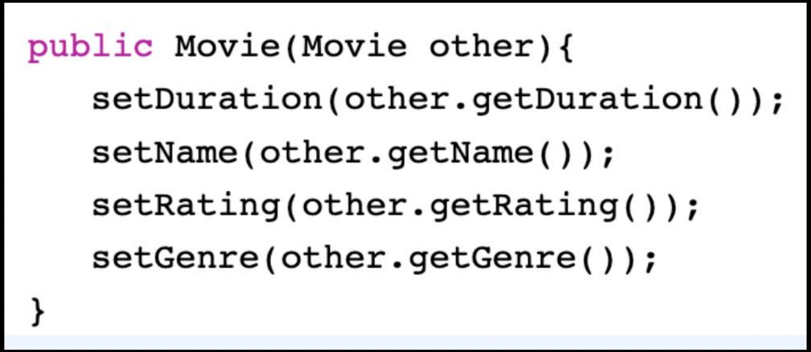
```java
public Movie(){
    duration = 100;
    name = "Untitled";
    rating = 0.0;
    genre = "Unclassified";
}
```

(C) Parameterized Constructor

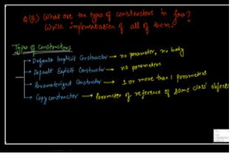


```java
public Movie(int duration, String name,
             double rating, String genre){

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

(D) Copy Constructor



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



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

Type 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

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 $\Rightarrow$ to resolve ambiguity and match function call to a particular function definition

- $\to$ Upcasting ✓
- $\to$ Downcasting ✗

$$\begin{tikzcd}
 & \text{(1) byte} \arrow[d] & \\
 & \text{(2) short} \arrow[d] & \\
\text{(2) char} \arrow[r] & \text{int (4)} \arrow[r] \arrow[d] \arrow[rd] & \text{float (4)} \arrow[d] \\
 & \text{(8) long} \arrow[r] \arrow[ru] & \text{double (8)}
\end{tikzcd}$$

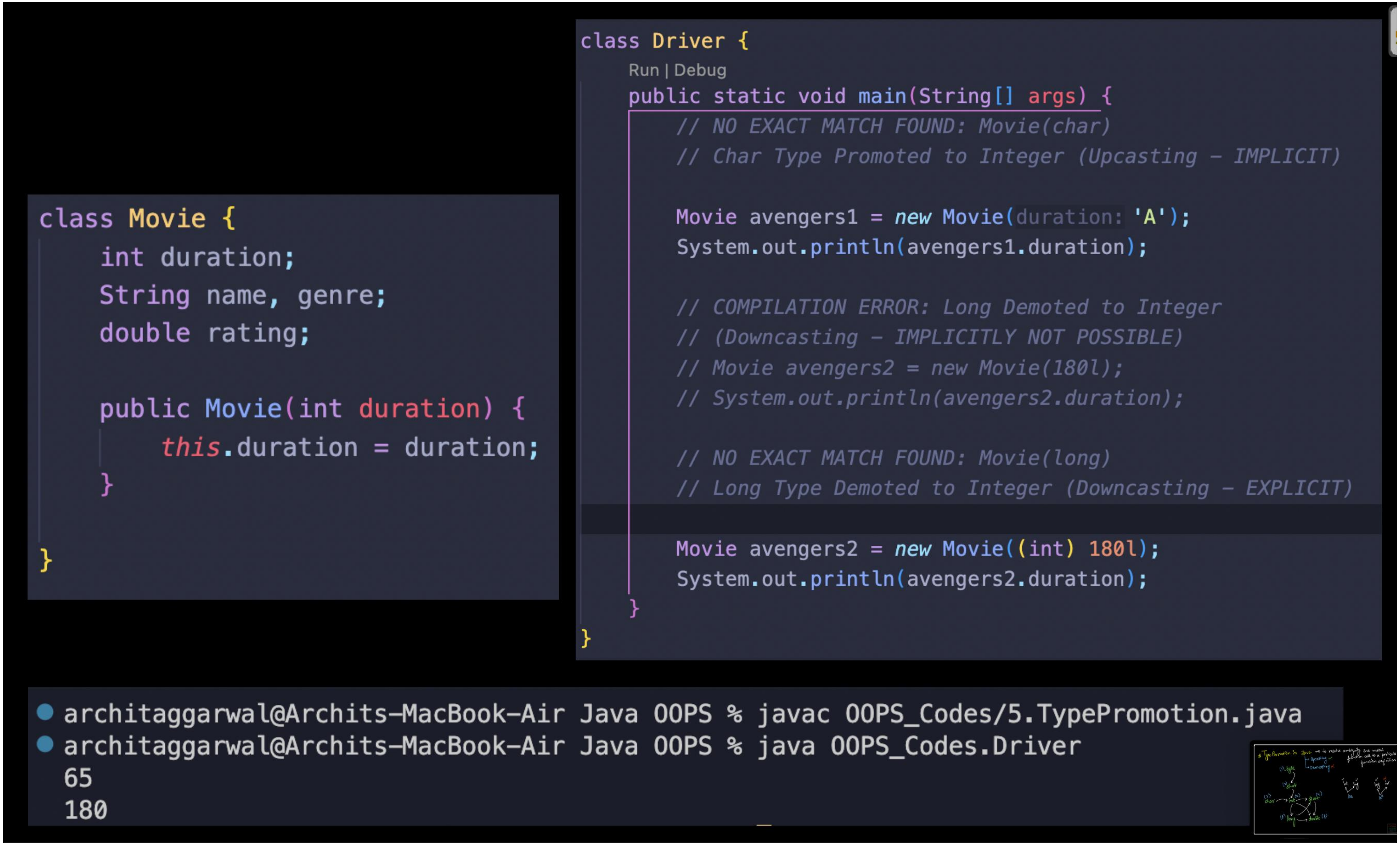
$$\begin{tikzcd}
\overset{\checkmark}{\text{int}} & & \overset{\checkmark}{\text{long}} & & \overset{\checkmark}{\text{long}} & & \overset{\times}{\text{int}} \\
 & 100 \arrow[lu] \arrow[ru] & & & & 10^{16} \arrow[lu] \arrow[ru] &
\end{tikzcd}$$

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
  - → Type of arguments are different
  - → Order of arguments are different

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



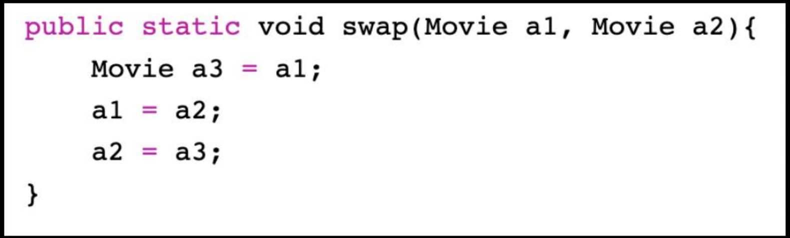
```java
class Movie {
    int duration;
    String name, genre;
    double rating;

    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);
    }
}
```

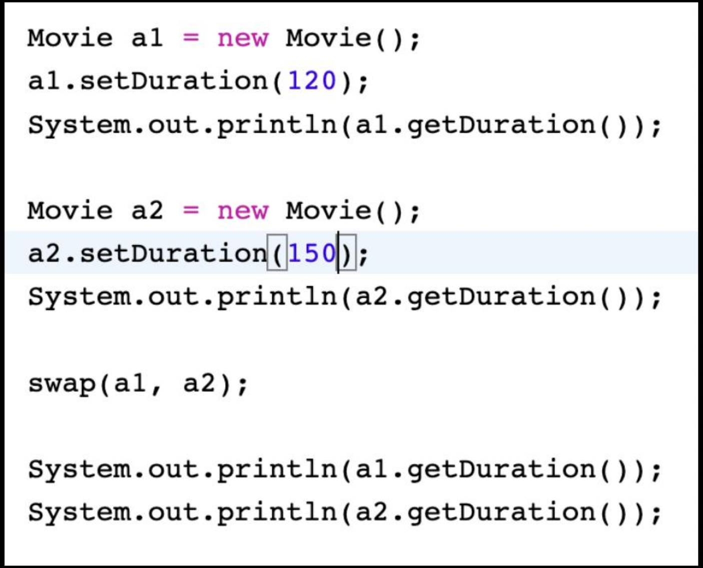
```
● architaggarwal@Archits-MacBook-Air Java OOPS % javac OOPS_Codes/5.TypePromotion.java
● architaggarwal@Archits-MacBook-Air Java OOPS % java OOPS_Codes.Driver
  65
  180
```

Q) Ⓐ 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

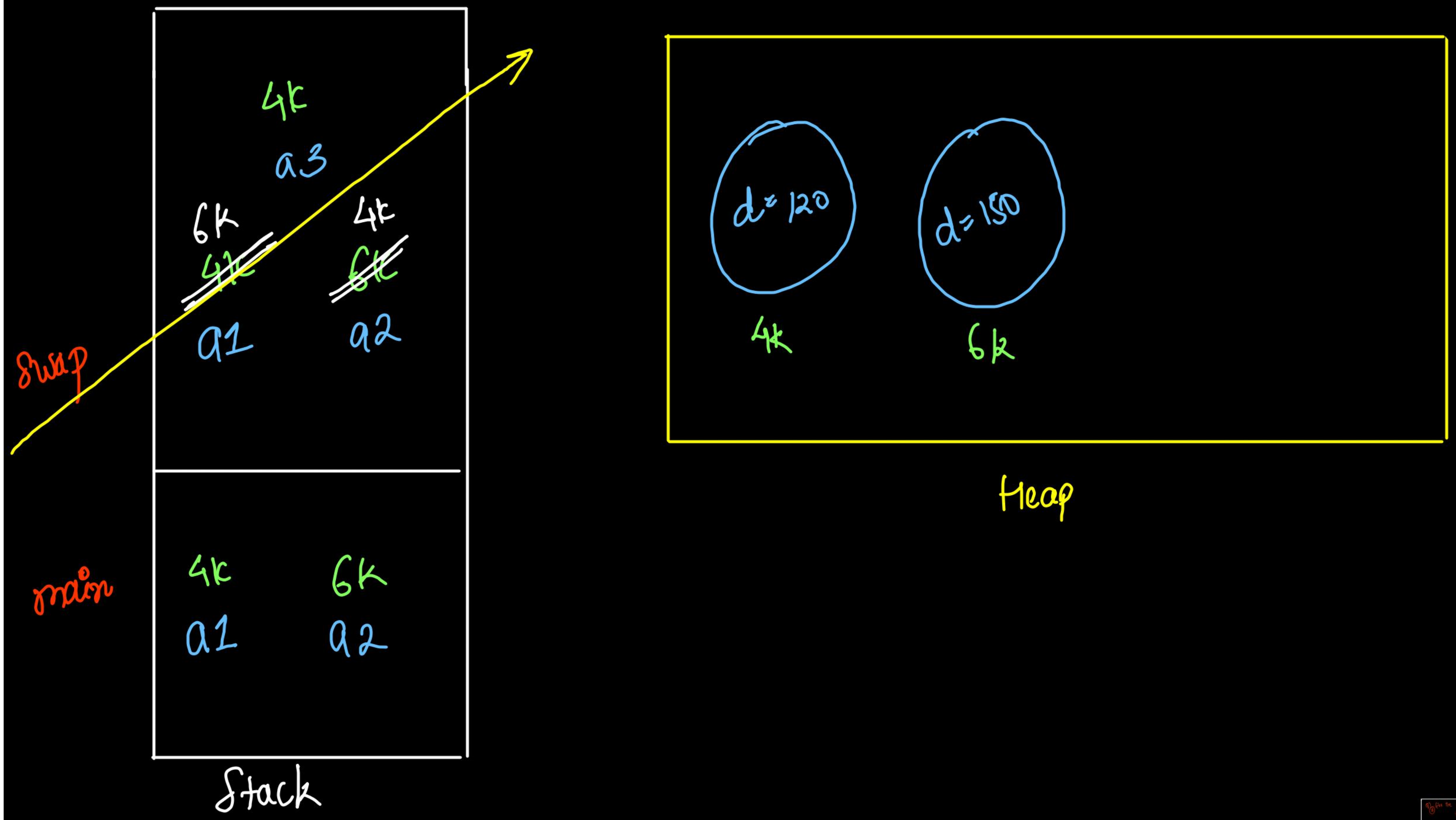
(c) 120, 150, 120, 120

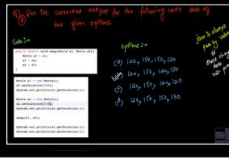
(d) 120, 150, 150, 150

java is always pass by value!

⇓

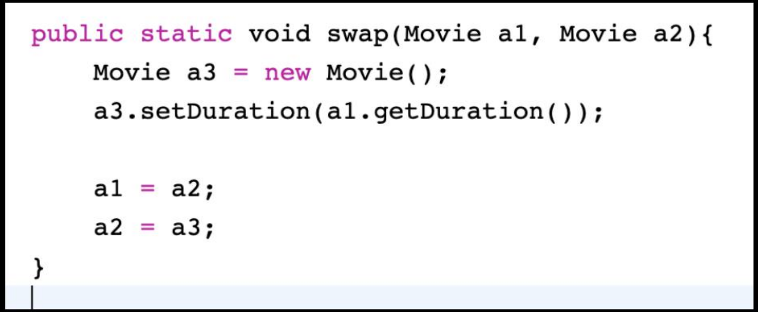
stack changes does not persist!


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

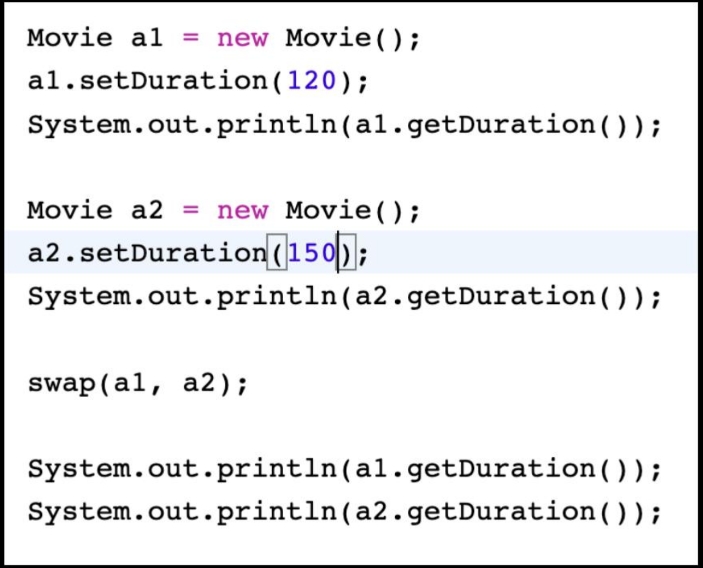


Q) Give the corrected output for the following code out of the given options.
$\textcircled{B}$

Code :→



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

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



```java
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());
```

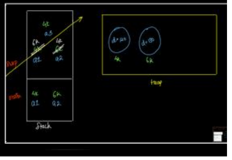
options :→

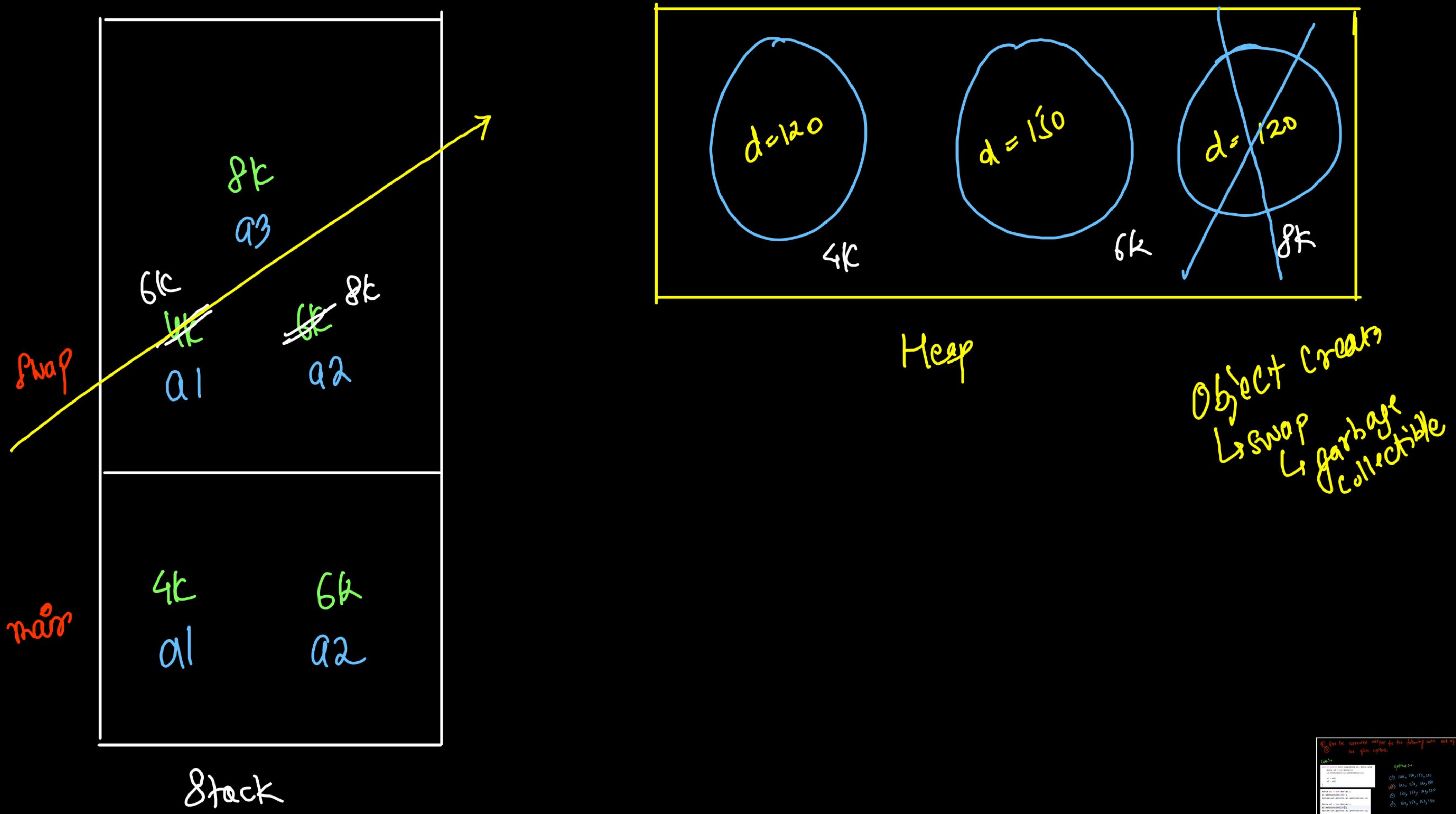
(a) 120, 150, 150, 120

[✓ (b) 120, 150, 120, 150]

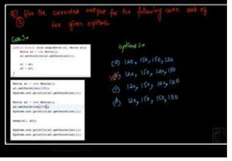
(c) 120, 150, 120, 120

(d) 120, 150, 150, 150



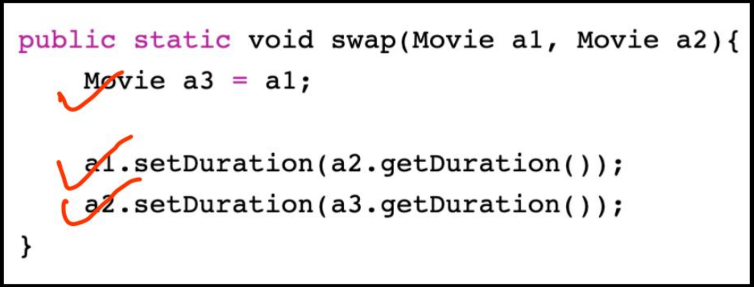


swap
main
8k
a3
6k
~~4k~~
a1
~~6k~~ 8k
a2
4k
a1
6k
a2
Stack
d=120
4k
d = 150
6k
d= 120
8k
Heap
Object Creation
↳ swap
↳ garbage collectible

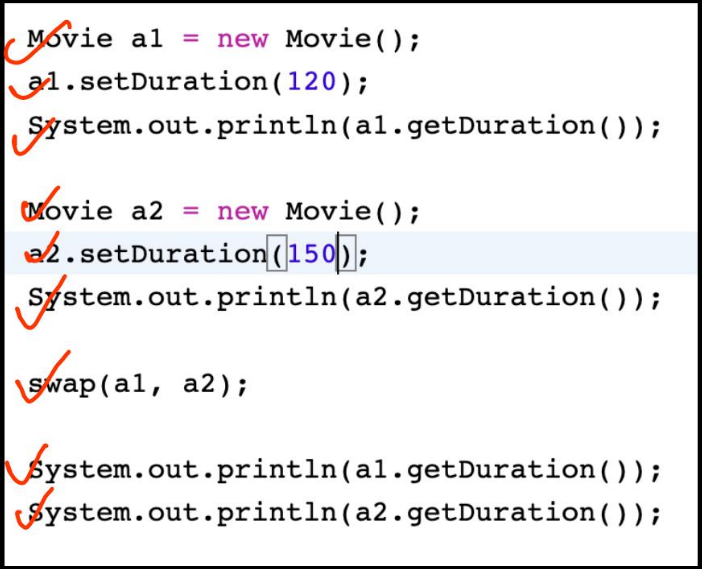


Q) Give the corrected output for the following code out of the given options.
Ⓒ

Code :→



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

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



```java
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());
```

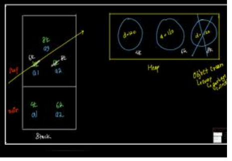
options :→

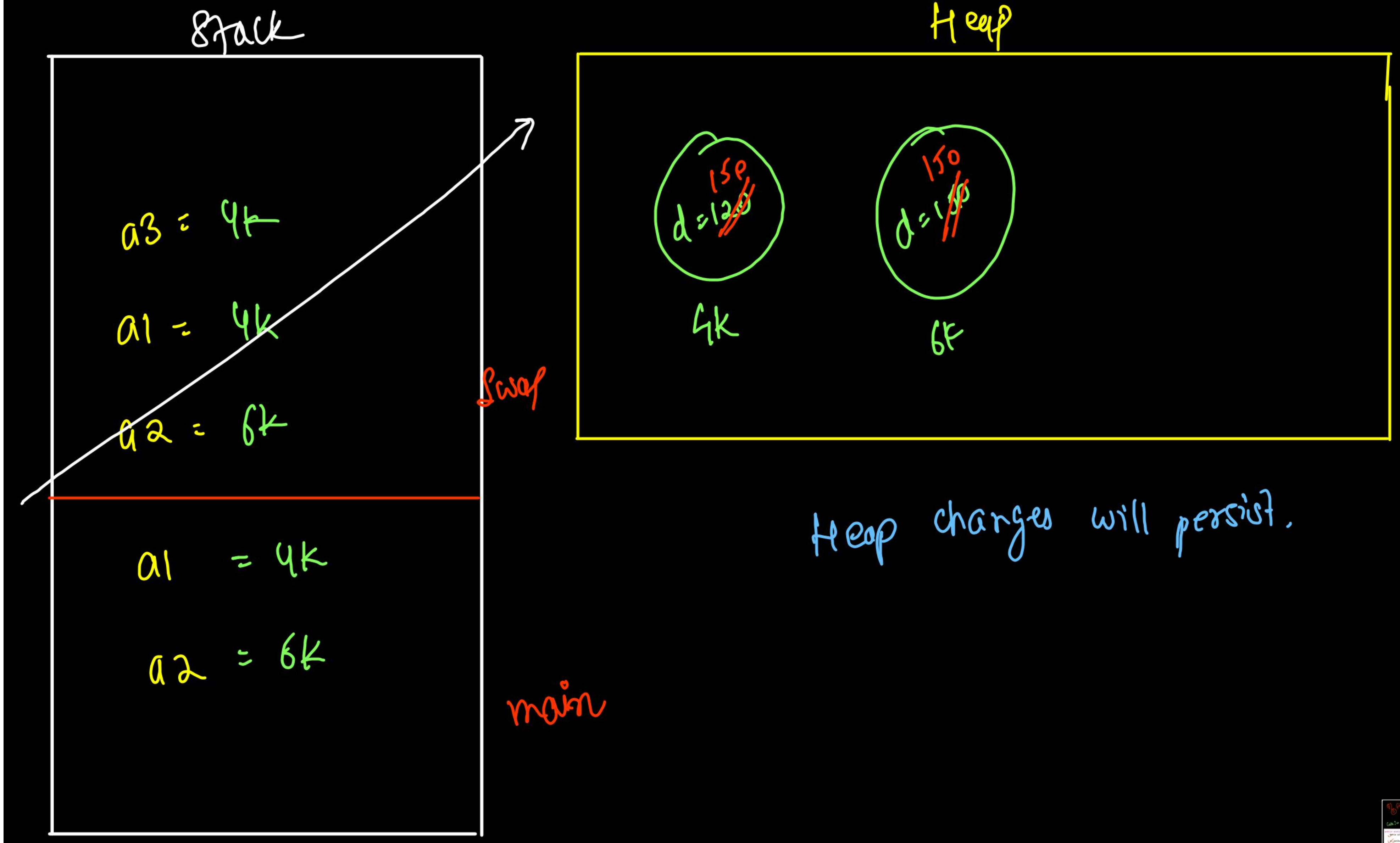
(a) 120, 150, 150, 120

(b) 120, 150, 120, 150

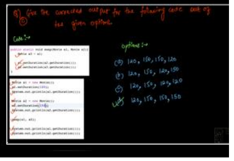
(c) 120, 150, 120, 120

[✓ (d)] 120, 150, 150, 150



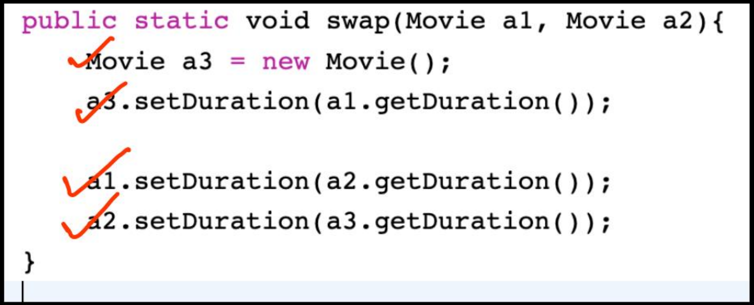


Heap changes will persist.

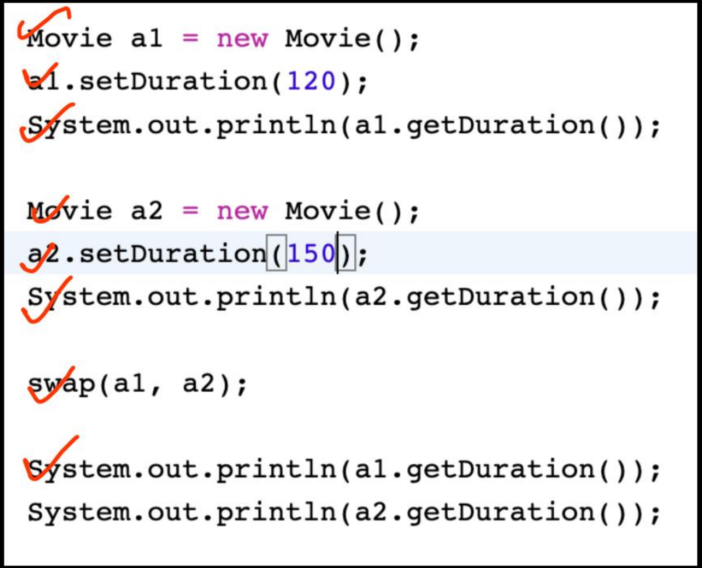


Q) Give the corrected output for the following code out of the given options.
[✓ D]

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());
```

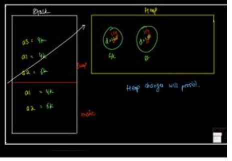
options :→

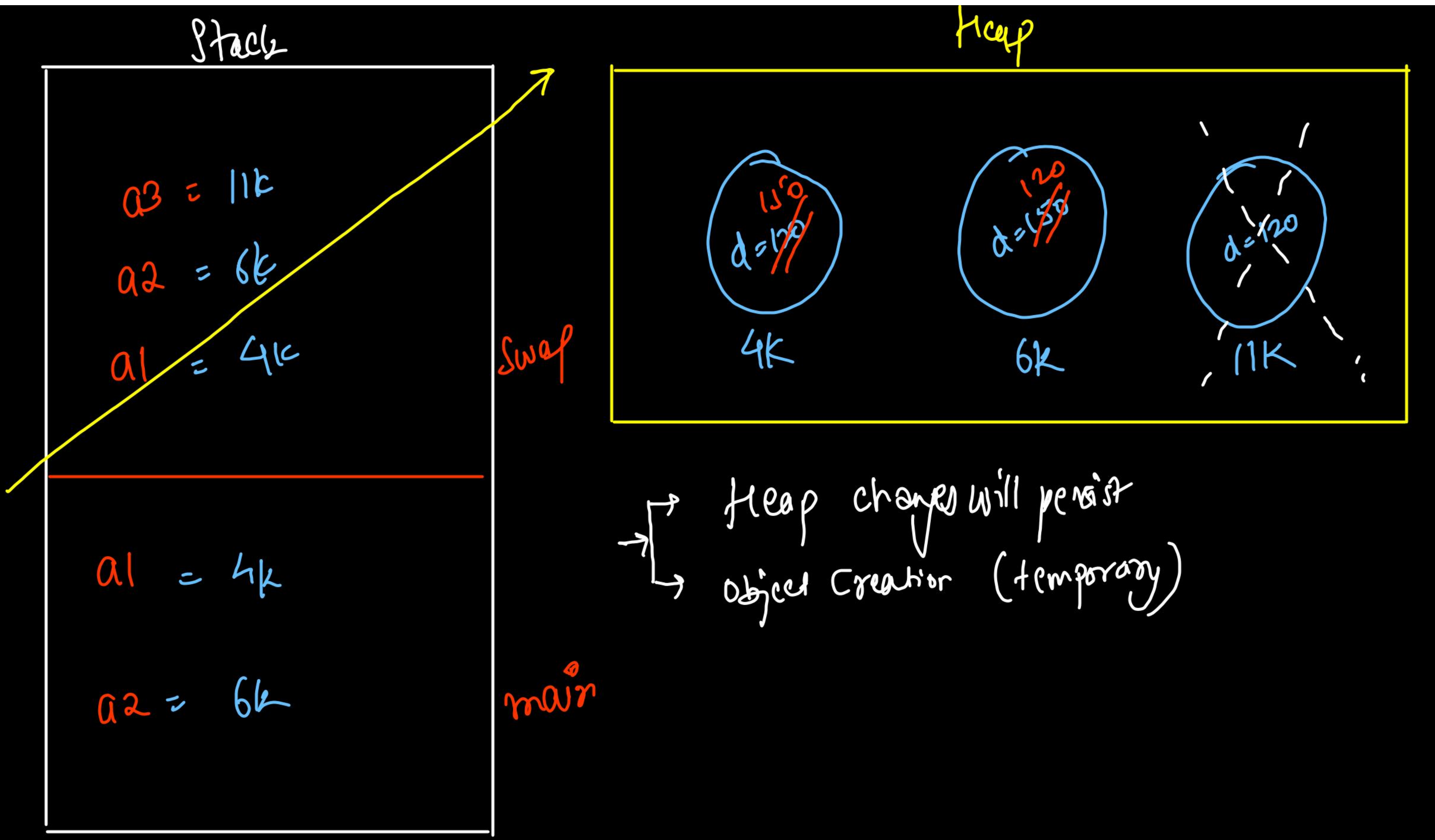
~~(a)~~ 120, 150, 150, 120

(b) 120, 150, 120, 150

(c) 120, 150, 120, 120

(d) 120, 150, 150, 150





→ Heap changes will persist

→ object creation (temporary)