Wrapper classes

1. Check if character is a Digit
2. Compare two Strings
3. Convert using valueof method
4. Create Boolean Wrapper usage
5. Convert null to wrapper classes

Pass by value and pass by reference

1. Write a program where a method accepts an integer parameter and tries to change its value. Print the value before and after the method call.
2. Create a method that takes two integer values and swaps them. Show that the original values remain unchanged after the method call.
3. Write a Java program to pass primitive data types to a method and observe whether changes inside the method affect the original variables.

```
package Assignment_Day7;

public class Problem1 {
public static void main(String[] args) {
char c = '7';
boolean isDigit = Character.isDigit(c);
System.out.println("Is '" + c + "' a digit? " + isDigit);
// two Strings
String str1 = "Hello";
String str2 = "World";
int comparison = str1.compareTo(str2);
```

```
System.out.println("Comparison result: " + comparison);
// valueOf
Integer intWrapper = Integer.valueOf("123");
System.out.println("String to Integer: " + intWrapper);
// Boolean Wrapper
Boolean boolWrapper = Boolean.valueOf("true");
System.out.println("Boolean wrapper: " + boolWrapper);
// null to wrapper
Integer nullWrapper = null;
System.out.println("Null wrapper: " + nullWrapper);
// Pass by value
System.out.println("\n=== Pass by Value Examples ===");
int num = 10;
System.out.println("Before method call: " + num);
tryToChange(num);
System.out.println("After method call: " + num);
// 2. Swap two integers
int a = 5, b = 10;
System.out.println("\nBefore swap: a = " + a + ", b = " + b);
swap(a, b);
System.out.println("After swap: a = " + a + ", b = " + b);
int x = 100;
System.out.println("\nBefore method: x = " + x);
modifyPrimitive(x);
System.out.println("After method: x = " + x);
}
public static void tryToChange(int value) {
value = 20;
System.out.println("Inside method: " + value);
}
public static void swap(int first, int second) {
int temp = first;
first = second;
second = temp;
System.out.println("Inside swap: first = " + first + ", second = " + second);
}
public static void modifyPrimitive(int num) {
num *= 2;
System.out.println("Inside method - modified: " + num);
}
}
```

```
Is '7' a digit? true
Comparison result: -15
String to Integer: 123
Boolean wrapper: true
Null wrapper: null

=== Pass by Value Examples ===
Before method call: 10
Inside method: 20
After method call: 10

Before swap: a = 5, b = 10
Inside swap: first = 10, second = 5
After swap: a = 5, b = 10

Before method: x = 100
Inside method - modified: 200
After method: x = 100
```

4. Create a class Box with a variable length. Write a method that modifies the value of length by passing the Box object. Show that the original object is modified.

```
package Assignment_Day7;


public class BoxDemo {
public static void main(String[] args) {
Box myBox = new Box(10);
System.out.println("Original length: " + myBox.length);
modifyBoxLength(myBox, 25);
System.out.println("Length after modification: " + myBox.length);
}
public static void modifyBoxLength(Box box, int newLength) {
box.length = newLength;
System.out.println("Inside method - modified length: " + box.length);
}
}
```

```
class Box {
int length;
public Box(int length) {
this.length = length;
}
}
```

```
Original length: 10
Inside method - modified length: 25
Length after modification: 25
```

5. Write a Java program to pass an object to a method and modify its internal fields. Verify that the changes reflect outside the method.

```
package Assignment_Day7;

class Box {
int width;
int height;
int depth;

public Box(int width, int height, int depth) {
this.width = width;
this.height = height;
this.depth = depth;
}

public void displayDimensions() {
System.out.println("Width: " + width + ", Height: " + height + ", Depth: " + depth);
}
}

public class Problem2 {
public static void main(String[] args) {
Box myBox = new Box(10, 20, 30);
System.out.println("Original dimensions:");
myBox.displayDimensions();

modifyBox(myBox);

System.out.println("\nDimensions after modification:");
myBox.displayDimensions();
```

```
}

public static void modifyBox(Box box) {
System.out.println("\nInside modifyBox method - before changes:");
box.displayDimensions();

box.width *= 2;
box.height *= 2;
box.depth *= 2;

System.out.println("Inside modifyBox method - after changes:");
box.displayDimensions();
}
}
```

```
Original length: 10
Inside method - modified length: 25
Length after modification: 25
```

Create a class Student with name and marks. Write a method to update the marks of a student. Demonstrate the changes in the original object.

```
package Assignment_Day7;
class Student {
String name;
int marks;
Student(String name, int marks) {
this.name = name;
this.marks = marks;
}
void updateMarks(int newMarks) {
this.marks = newMarks;
}
}

public class Student_Demo {
public static void main(String[] args) {
Student s = new Student("John", 85);
System.out.println("Before: " + s.marks);
s.updateMarks(90);
System.out.println("After: " + s.marks);
```

```
}
}
```

```
Before: 85
After: 90
```

7. Create a program to show that Java is strictly "call by value" even when passing objects (object references are passed by value).

```
package Assignment_Day7;

class ValueDemo {
static void change(int x) {
x = 10;
}
public static void main(String[] args) {
int x = 5;
change(x);
System.out.println(x);
}
}
```

```
5
```

Write a program where you assign a new object to a reference passed into a method. Show that the original reference does not change.

```
package Assignment_Day7;

class RefDemo {
static void changeRef(Student s) {
s = new Student("Alice", 95);
```

```
}
public static void main(String[] args) {
Student s = new Student("Bob", 80);
changeRef(s);
System.out.println(s.name);
}
}
```

```
Bob
```

Explain the difference between passing primitive and non-primitive types to methods in Java with examples.

```
package Assignment_Day7;

class TypeDemo {
static void changePrimitive(int x) { x = 10; }
static void changeObject(Student s) { s.marks = 100; }
public static void main(String[] args) {
int x = 5;
Student s = new Student("Tom", 75);
changePrimitive(x);
changeObject(s);
System.out.println(x);
System.out.println(s.marks);
}
}
```

```
5
100
```

Can you simulate call by reference in Java using a wrapper class or array? Justify with a program.

```
package Assignment_Day7;

class IntWrapper { int value; }

class RefSimulation {
static void change(IntWrapper w) {
w.value = 10;
}
public static void main(String[] args) {
IntWrapper w = new IntWrapper();
```

```
w.value = 5;
change(w);
System.out.println(w.value);
}
}
```

```
5
100
```

1 Write a program to create a thread by extending the Thread class and print numbers from 1 to 5.

```
package Assignment_Day7;

class t1 extends Thread {
public void run() {
for(int i=1; i<=5; i++) {
System.out.println(i);
}
}
public static void main(String[] args) {
new t1().start();
}
}
```

```
1
2
3
4
5
```

2 Create a thread by implementing the Runnable interface that prints the current thread name.

```
package Assignment_Day7;
class t2 implements Runnable {
public void run() {
System.out.println(Thread.currentThread().getName());
}
public static void main(String[] args) {
new Thread(new t2()).start();
}
}
```

```
Thread-0
```

Write a program to create two threads, each printing a different message 5 times.

```
package Assignment_Day7;

class TwoThreads {
public static void main(String[] args) {
new Thread(() -> {
for(int i=0; i<5; i++) System.out.println("Hello");
}).start();
new Thread(() -> {
for(int i=0; i<5; i++) System.out.println("World");
}).start();
}
}
```

```
Hello
Hello
Hello
Hello
Hello
World
World
World
World
World
```

4 Demonstrate the use of Thread.sleep() by pausing execution between numbers from 1 to 3.

```
package Assignment_Day7;
class SleepDemo {
public static void main(String[] args) throws InterruptedException {
for(int i=1; i<=3; i++) {
System.out.println(i);
```

```
Thread.sleep(1000);
}
}
}
```

```
1
2
3
```

5 Create a thread and use Thread.yield() to pause and give chance to another thread.

```
package Assignment_Day7;

class YieldDemo {
public static void main(String[] args) {
new Thread(() -> {
for(int i=0; i<5; i++) {
System.out.println("Thread 1");
Thread.yield();
}
}).start();
new Thread(() -> {
for(int i=0; i<5; i++) {
System.out.println("Thread 2");
}
}).start();
}
}
```

```
Thread 1
Thread 1
Thread 1
Thread 1
Thread 1
Thread 2
Thread 2
Thread 2
Thread 2
Thread 2
```

6 Implement a program where two threads print even and odd numbers respectively.

```
package Assignment_Day7;

class EvenOdd {
public static void main(String[] args) {
new Thread(() -> {
for(int i=1; i<=10; i+=2) System.out.println("Odd: "+i);
}).start();
new Thread(() -> {
for(int i=2; i<=10; i+=2) System.out.println("Even: "+i);
}).start();
}
}
```

```
Even: 2
Even: 4
Even: 6
Even: 8
Even: 10
Odd: 1
Odd: 3
Odd: 5
Odd: 7
Odd: 9
```

7 Create a program that starts three threads and sets different priorities for them.

```
package Assignment_Day7;

class PriorityDemo {
public static void main(String[] args) {
Thread t1 = new Thread(() -> System.out.println("High priority"));
Thread t2 = new Thread(() -> System.out.println("Low priority"));
t1.setPriority(Thread.MAX_PRIORITY);
t2.setPriority(Thread.MIN_PRIORITY);
t1.start();
t2.start();
}
}
```

```
Low priority
High priority
```

8 Write a program to demonstrate Thread.join() – wait for a thread to finish before proceeding.

```
package Assignment_Day7;

class JoinDemo {
public static void main(String[] args) throws InterruptedException {
Thread t = new Thread(() -> System.out.println("Thread running"));
t.start();
t.join();
System.out.println("Main thread continues");
}
}
```

```
Thread running
Main thread continues
```

9 Show how to stop a thread using a boolean flag.

```
package Assignment_Day7;

class StoppableThread extends Thread {
private boolean stop = false;
public void run() {
while(!stop) {
System.out.println("Running...");
}
}
public void stopThread() { stop = true; }
public static void main(String[] args) throws InterruptedException {
StoppableThread t = new StoppableThread();
t.start();
Thread.sleep(100);
t.stopThread();
}
}
```

```
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
Running...
```

10 Create a program with multiple threads that access a shared counter without synchronization. Show the race condition.

```
package Assignment_Day7;

class Counter {
static int count = 0;
public static void main(String[] args) {
Runnable r = () -> {
for(int i=0; i<1000; i++) count++;
};
new Thread(r).start();
new Thread(r).start();
try { Thread.sleep(1000); }
catch (InterruptedException e) {}
System.out.println(count);
}
}
```

```
1894
```

11 Solve the above problem using synchronized keyword to prevent race condition.

```
package Assignment_Day7;

class SyncCounter {
static int count = 0;
synchronized static void increment() { count++; }
public static void main(String[] args) throws InterruptedException {
Thread t1 = new Thread(() -> {
for(int i=0; i<1000; i++) increment();
});
Thread t2 = new Thread(() -> {
for(int i=0; i<1000; i++) increment();
});
t1.start(); t2.start();
t1.join(); t2.join();
}
}
```

```
2000
```

12 Write a Java program using synchronized block to ensure mutual exclusion.

```
class Counter {
int count = 0;
final Object lock = new Object();

void increment() {
synchronized(lock) {
count++;
}
}

public static void main(String[] args) throws InterruptedException {
Counter c = new Counter();
Thread t1 = new Thread(() -> {
for(int i=0; i<1000; i++) c.increment();
});
Thread t2 = new Thread(() -> {
for(int i=0; i<1000; i++) c.increment();
});
t1.start(); t2.start();
t1.join(); t2.join();
System.out.println("Count: " + c.count); // Always 2000
}
```

```
}
Count: 2000
```

13 Implement a BankAccount class accessed by multiple threads to deposit and withdraw money. Use synchronization.

```
package Assignment_Day7;

class BankAccount {
private int balance = 1000;
synchronized void deposit(int amount) {
balance += amount;
System.out.println("Deposited " + amount + ", new balance: " + balance);
}
synchronized void withdraw(int amount) {
if(balance >= amount) {
balance -= amount;
System.out.println("Withdrew " + amount + ", new balance: " + balance);
}
}
public static void main(String[] args) {
BankAccount account = new BankAccount();
new Thread(() -> account.deposit(500)).start();
new Thread(() -> account.withdraw(200)).start();
}
}
```

```
Deposited 500, new balance: 1500
Withdrew 200, new balance: 1300
```

15 Create a program where one thread prints A-Z and another prints 1-26 alternately.

```
package Assignment_Day7;

class AlternatePrint {
boolean isLetterTurn = true;
synchronized void printLetter() throws InterruptedException {
for(char c='A'; c<='Z'; c++) {
while(!isLetterTurn) wait();
System.out.print(c + " ");
isLetterTurn = false;
```

```
notify();
}
}
synchronized void printNumber() throws InterruptedException {
for(int i=1; i<=26; i++) {
while(isLetterTurn) wait();
System.out.print(i + " ");
isLetterTurn = true;
notify();
}
}
public static void main(String[] args) {
AlternatePrint ap = new AlternatePrint();
new Thread(() -> { try { ap.printLetter(); } catch(Exception e) {} }).start();
new Thread(() -> { try { ap.printNumber(); } catch(Exception e) {} }).start();
}
}
```

```
A 1 B 2 C 3 D 4 E 5 F 6 G 7 H 8 I 9 J 10 K 11 L 12 M 13 N 14 O 15 P 16 Q 17 R 18 S 19 T 20 U 21 V 22 W 23 X 24 Y 25 Z 26
```

16 Write a program that demonstrates inter-thread communication using wait() and notifyAll().

```
package Assignment_Day7;

class WaitNotifyAll {
boolean condition = false;
synchronized void waitForCondition() throws InterruptedException {
System.out.println("Waiting for condition...");
while(!condition) wait();
System.out.println("Condition met!");
}
synchronized void setCondition() {
condition = true;
notifyAll();
}
public static void main(String[] args) {
WaitNotifyAll wna = new WaitNotifyAll();
for(int i=0; i<3; i++) {
new Thread(() -> { try { wna.waitForCondition(); } catch(Exception e) {} }).start();
}
new Thread(() -> wna.setCondition()).start();
}
}
```

```
Waiting for condition...
Waiting for condition...
Waiting for condition...
Condition met!
Condition met!
Condition met!
```

17 Create a daemon thread that runs in background and prints time every second.

```
package Assignment_Day7;

class DaemonThread {
public static void main(String[] args) {
Thread daemon = new Thread(() -> {
while(true) {
System.out.println("Current time: " + System.currentTimeMillis());
try { Thread.sleep(1000); } catch(Exception e) {}
}
});
daemon.setDaemon(true);
daemon.start();
System.out.println("Main thread exiting...");
}
}
```

```
Main thread exiting...
```

18 Demonstrate the use of Thread.isAlive() to check thread status.

```
package Assignment_Day7;
class ThreadState{
public static void main(String[] args) throws InterruptedException {
Thread t = new Thread(() -> {
try { Thread.sleep(2000); } catch(Exception e) {}
});
System.out.println("Before start: " + t.isAlive());
t.start();
System.out.println("After start: " + t.isAlive());
t.join();
System.out.println("After join: " + t.isAlive());
```

```
}
}
```

```
Before start: false
After start: true
After join: false
```

19 Write a program to demonstrate thread group creation and management.

```
package Assignment_Day7;

class tg {
public static void main(String[] args) {
ThreadGroup group = new ThreadGroup("MyGroup");
Thread t1 = new Thread(group, () -> System.out.println("Thread 1"));
Thread t2 = new Thread(group, () -> System.out.println("Thread 2"));
t1.start();
t2.start();
System.out.println("Active threads: " + group.activeCount());
}
}
```

```
Thread 2
Thread 1
Active threads: 0
```

20 Create a thread that performs a simple task (like multiplication) and returns result using Callable and Future.

```
package Assignment_Day7;

import java.util.concurrent.*;

class cd {
public static void main(String[] args) throws Exception {
ExecutorService executor = Executors.newSingleThreadExecutor();
Future<Integer> future = executor.submit(() -> {
Thread.sleep(1000);
return 5 * 7;
});
System.out.println("Result: " + future.get());
executor.shutdown();
```

```
}
}
```

```
Result: 35
```