Day7\_Assignment\_Answers:

1. package day\_7\_Assignment;

/\*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\*/

public class Que1 {

public static void main(String[] args) {

char ch='1';//1

System.***out***.println(Character.*isDigit*(ch));

String s1="Hello";

String s2="Hel";

System.***out***.println(s1.compareTo(s2));//2

String s="234";

System.***out***.println(s.*valueOf*(s));//3

boolean b=true;

Boolean b1=b;

System.***out***.println(b1);//4

Integer s3=(Integer)null;

System.***out***.println(s3);//5

}

}

Output:

true

2

234

true

null

2. package day\_7\_Assignment;

/\*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.

\*/

public class Que2 {

public void change(int n) {

n=20;

}

public static void main(String[] args) {

// **TODO** Auto-generated method stub

int n=10;

Que2 q=new Que2();

System.***out***.println(n);

q.change(n);

System.***out***.println(n);

}

}

Output:

10

10

3. package day\_7\_Assignment;

/\*Create a method that takes two integer values and swaps them.

\* Show that the original values remain unchanged after the method call.

\*/

public class Que3 {

public void swap(int n1,int n2) {

n1=n1+n2;

n2=n1-n2;

n1=n1-n2;

}

public static void main(String[] args) {

// **TODO** Auto-generated method stub

Que3 q=new Que3();

int n1=10;

int n2=20;

System.***out***.println(n1+" " +n2);

q.swap(n1, n2);

System.***out***.println(n1+" " +n2);

}

}

Output:

10 20

10 20

4. package day\_7\_Assignment;

/\*3.Write a Java program to pass primitive data types to a method

\* and observe whether changes inside the method affect the original variables.

\*/

public class Que4 {

public void passPrimitive(int a) {

a=80;

}

public static void main(String[] args) {

// **TODO** Auto-generated method stub

Que4 q=new Que4();

int a=9;

q.passPrimitive(a);

System.***out***.println(a);

}

}

Output:

9

5. package day\_7\_Assignment;

/\*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.\*/

public class Que5 {

int length;

public void modify(Que5 q) {

q.length=40;

}

public static void main(String[] args) {

// **TODO** Auto-generated method stub

Que5 q=new Que5();

q.length=80;

q.modify(q);

System.***out***.println(q.length);

}

}

Output:

40

6. package day\_7\_Assignment;

class Test extends Thread{

volatile boolean flag=true;

public void stopping() {

flag=false;

}

public void run() {

while(flag==true) {

System.***out***.println("thread started");

try {

Thread.*sleep*(2000);

}catch(Exception e) {

System.***out***.println(e);

}

}

System.***out***.println("Thread stopped execution");

}

}

public class Thread\_stop {

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

// **TODO** Auto-generated method stub

Test t=new Test();

t.start();

Thread.*sleep*(5000);

t.stopping();

}

}

Output:

thread started

thread started

thread started

Thread stopped execution

7. package day\_7\_Assignment;

/\*7.Create a program to show that Java is strictly "call by value"

\* even when passing objects (object references are passed by value).\*/

public class Pass\_By\_Value {

int length;

public void modify(Pass\_By\_Value q) {

q.length=40;

}

public static void main(String[] args) {

// **TODO** Auto-generated method stub

Pass\_By\_Value q=new Pass\_By\_Value();

q.length=80;

q.modify(q);/\*does't affect the original Reference ,

b/c only a copy of the reference is passed,& changes to that copy don't affect the original\*/

System.***out***.println(q.length);

}

}

Output:

40

8. package day\_7\_Assignment;

/\*8.Write a program where you assign a new object to a reference passed into a method.

\* Show that the original reference does not change.

\*/

public class Pass\_By\_Value2 {

int length;

public void modify(Pass\_By\_Value q) {

q.length=40;

}

public static void main(String[] args) {

// **TODO** Auto-generated method stub

Pass\_By\_Value q=new Pass\_By\_Value();

q.length=80;

q.modify(q);

System.***out***.println(q.length);

Pass\_By\_Value q1=new Pass\_By\_Value();

System.***out***.println(q1.length);//the original reference does not change.

}

}

Output:

40

0

9. import java.util.concurrent.\*;

public class MultiplyTask implements Callable<Integer> {

private int a, b;

public MultiplyTask(int a, int b) {

this.a = a;

this.b = b;

}

@Override

public Integer call() {

return a \* b;

}

public static void main(String[] args) {

ExecutorService executor = Executors.newSingleThreadExecutor();

MultiplyTask task = new MultiplyTask(7, 6);

Future<Integer> future = executor.submit(task);

try {

int result = future.get(); // Blocks until the result is available

System.out.println("Multiplication Result: " + result);

} catch (InterruptedException | ExecutionException e) {

e.printStackTrace();

} finally {

executor.shutdown();

}

}

}

Output:

Multiplication Result: 42

10. public class ThreadGroupDemo {

public static void main(String[] args) {

ThreadGroup group = new ThreadGroup("Group");

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());

group.list();

}

}

Output:

Thread 1

Thread 2

Active threads: 2

java.lang.ThreadGroup[name=Group,maxpri=10]

Thread[Thread 1,5,Group]

Thread[Thread 2,5,Group]

11. public class WaitNotifyDemo {

private static final Object lock = new Object();

public static void main(String[] args) {

Runnable producer = () -> {

synchronized (lock) {

System.out.println("Produced");

lock.notifyAll();

}

};

Runnable consumer = () -> {

synchronized (lock) {

try {

lock.wait();

System.out.println("Consumed");

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

};

new Thread(producer).start();

new Thread(consumer).start();

}

}

Output:

Produced

Consumed

12. public class AlternatePrint {

private static final Object lock = new Object();

private static boolean isLetterTurn = true;

public static void main(String[] args) {

Thread letterThread = new Thread(() -> {

for (char c = 'A'; c <= 'Z'; c++) {

synchronized (lock) {

while (!isLetterTurn) {

try {

lock.wait();

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

System.out.print(c);

isLetterTurn = false;

lock.notify();

}

}

});

Thread numberThread = new Thread(() -> {

for (int i = 1; i <= 26; i++) {

synchronized (lock) {

while (isLetterTurn) {

try {

lock.wait();

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

System.out.print(i);

isLetterTurn = true;

lock.notify();

}

}

});

letterThread.start();

numberThread.start();

}

}

Output:

A1B2C3D4E5F6G7H8I9J10K11L12M13N14O15P16Q17R18S19T20U21V22W23X24Y25Z26

13. public class BankAccount {

private int balance = 1000;

public synchronized void deposit(int amount) {

balance += amount;

System.out.println("Deposited: " + amount + ", Balance: " + balance);

}

public synchronized void withdraw(int amount) {

if (balance >= amount) {

balance -= amount;

System.out.println("Withdrawn: " + amount + ", Balance: " + balance);

} else {

System.out.println("Insufficient funds to withdraw: " + amount);

}

}

public int getBalance() {

return balance;

}

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

BankAccount account = new BankAccount();

Thread t1 = new Thread(() -> {

for (int i = 0; i < 3; i++) {

account.deposit(200);

try { Thread.sleep(50); } catch (InterruptedException e) { Thread.currentThread().interrupt(); }

}

});

Thread t2 = new Thread(() -> {

for (int i = 0; i < 3; i++) {

account.withdraw(100);

try { Thread.sleep(100); } catch (InterruptedException e) { Thread.currentThread().interrupt(); }

}

});

t1.start();

t2.start();

t1.join();

t2.join();

System.out.println("Final Balance: " + account.getBalance());

}

}

Output:

Deposited: 200, Balance: 1200

Withdrawn: 100, Balance: 1100

Deposited: 200, Balance: 1300

Withdrawn: 100, Balance: 1200

Deposited: 200, Balance: 1400

Withdrawn: 100, Balance: 1300

Final Balance: 1300

14. public class Counter {

private int count;

private final Object lock = new Object();

public void increment() {

synchronized (lock) {

count++;

}

}

public int getCount() {

synchronized (lock) {

return count;

}

}

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

Counter c = new Counter();

Runnable task = () -> { for (int i = 0; i < 1000; i++) c.increment(); };

Thread t1 = new Thread(task);

Thread t2 = new Thread(task);

t1.start();

t2.start();

t1.join();

t2.join();

System.out.println("Final count: " + c.getCount());

}

}

Output:

Final count: 2000

15. public class RaceConditionDemo {

private static int count = 0;

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

Runnable task = () -> {

for (int i = 0; i < 1000; i++) {

count++; // not atomic

}

};

Thread t1 = new Thread(task);

Thread t2 = new Thread(task);

t1.start();

t2.start();

t1.join();

t2.join();

System.out.println("Final count: " + count);

}

}

Output:

Final count: 1785