THROWS :- It is a keyword which declares to the method that which exception to be thrown.

SYNTAX :- throws

THROW :- It is a keyword of JAVA which throws exceptional object by explicit

MULTI TASKING / MULTI PROCESSOR :- Utilizing ideal time of µ-Processor is called multi tasking (or) multi processor.

THREAD :- A thread is an independent process. A thread can be created in JAVA by 2 ways

1. By extending Thread class
2. By implementing runnable interface

New -> Class -> Thread 2 -> Browse -> Thread class -> FINISH

RIGHT CLICK -> @ override | run Method

public void run () {

Browse -> thread.java.lan

package

while creating a class

RIGHT CLICK -> SOURCE -> OVERRIDE / IMPLEMENT METHODS -> SEARCH RUN

Thread1.java

→
```
package sreeja;
public class Thread1 extends Thread
{
  public void run()
  {
    for(int i=1; i<=10; i+=2)
    {
      System.out.println(i);
      try{
        Thread.sleep(1000);
      }
      catch(InterruptedException e)
      {
        e.printStackTrace();
      }
    }
  }
}
```

Thread2.java

→
```
package sreeja;
public class Thread2 extends Thread
{
  public void run()
  {
    for(int i=2; i<=10; i+=2)
    {
      System.out.println(i);
```

```
try {
    Thread.sleep(1000);
  }
  catch (InterruptedException e)
  {
    e.printStackTrace();
  }
 }
}
}
```

RUN() → RUNABLE INTERFACE → ABSTRACT

Demo.java

```
package sreeja;

public class Demo
{
  public static void main(String arr[])
  {
    Thread1 p = new Thread1();
    Thread2 q = new Thread2();
    p.start(); q.start();
  }
}
```

O/P:-

1
2
3
4
5
6
7
8
9
10

RUN()

↓

RUNABLE

INTERFACE

↓

ABSTRACT

03-07-2024

JOIN() :- Join() waits parent thread until child thread dies.

Thread-1.java

~~Thread-1.java~~

```
-> package seeja;

public class Thread1 implements Runnable {
  @override
  public void run()
  {
    for(int i=1; i<=10; i+=2)
    {
      System.out.println(i);
      try
      {
        Thread.sleep(1000);
      }
      catch (InterruptedException e)
      {
        e.printStackTrace();
      }
    }
  }
}
```

Thread1_1.java

1st write till implements Runnable → It gives error → CLICK ON IT → ADD UNIMPLEMENTED METHODS

it displays @override

```
public void run()
{
}
```

RUN()
↓
RUNNABLE INTERFACE
↓
ABSTRACT

Thread - 2. java

Thread1_2. java

-> package sraja;

public class thread2 implements Runnable

{

@Override

public void run()

{

for(int i = 2; i <= 10; i += 2)

{

System.out.println(i);

try{

Thread.sleep(1000);

}

catch (InterruptedException e)

{

e.printStackTrace();

}

}

}

int i=2;

1st write till implements Runnable.

↓

It gives error

↓

Click On IT

↓

Add Unimplemented methods

↓

it displays

@Override

public void run()

{

}

Demo. java

-> package sraja;

Thread1_2_Combine. java

public class Demo{

public static void main(String ar[]) throws InterruptedException

{

Thread1 p = new Thread1();

Thread2 q = new Thread2();

Thread r = new Thread(p);

Thread s = new Thread(q);

r.start(), s.start();

r.join(), s.join();

System.out.println("hello");

}}

O/P:- COMBINATION OF Thread1 Thread2

2
1
4
3
6
5
8
7
10
9
hello

Daemon() Thread = service thread.

Demo.java

Daemon.java

```
package sreeja

public class Demo{

    public static void main(String ar[])

    {

        ThreadGroup tg = new ThreadGroup("satvik");

            Thread t = new Thread(tg, "hello");

        System.out.println(t.getName());

        t.setName("Hemant");

        System.out.println(t.getName());

        system.out.println(t.getPriority());

        t.setPriority(Thread.MAX_PRIORITY);

        System.                    (t.get is

        System.out.println(t.getPriority);

        System.out.println(t.isAlive());

        t.start();

        System.out.println(t.isAlive());

        System.out.println(t.isDaemon());

        t.setDaemon(true);

        System.out.println(t.isDaemon());

        System.out.println(t.getThreadGroup());

    }

}
```

O/P:-

2
hello
Hemant
5
10
false

true
false
true
null