**Inner Classes – Part-03**

* **Anonymous Inner Classes:**

Sometimes we can declare inner-class without name. Such type of inner-classes is called, anonymous inner classes.

Note: Recall about asking the address from anonymous person. Or getting a ticket from a conductor.

The main purpose of anonymous inner classes is just for instant use (one-time usage).

Based on declaration and behavior there are three types of anonymous inner classes

1. Anonymous inner class that extends a class.
2. Anonymous inner class that implements an interface.
3. Anonymous inner class that defined inside arguments.

* **Anonymous inner class that extends a class:**

Syntax:

PopCorn p = new PopCorn(){

};

We have to read it from left to right like below:

A class which extends PopCorn without name and creating a child object (new PopCorn) and assigning it to Parent class reference.

Note: Eventhough we call (new PopCorn()) it is not the parent class, it is the chid class.

**Example:**

class PopCorn{

public void taste(){

System.out.println(“salty”);

}

}

class Test{

public static void main(String[] args){

PopCorn p = new PopCorn(){

public void taste(){

System.out.println(“spicy”);

}

};

p.taste();

PopCorn p1 = new PopCorn();

p1.taste();

PopCorn p2 = new PopCorn(){

public void taste(){

System.out.println(“Sweet”);

}

};

p2.taste();

System.out.println(p.getClass().getName()); Test$1

System.out.println(p2.getClass().getName()); PopCorn

System.out.println(p3.getClass().getName()); Test$2

}

}

The generated .class files are:

PopCorn.class

Test.class

Test$1.class

Test$2.class

* **Analysis:**

1. PopCorn p = new PopCorn();

Just we are creating PopCorn object.

1. PopCorn p = new PopCorn(){

};

We are declaring a class that extends PopCorn without name (anonymous inner class).

For that child class we are creating an object with parent reference.

1. PopCorn p = new PopCorn(){

public void taste(){

System.out.println(“Salt”);

}

};

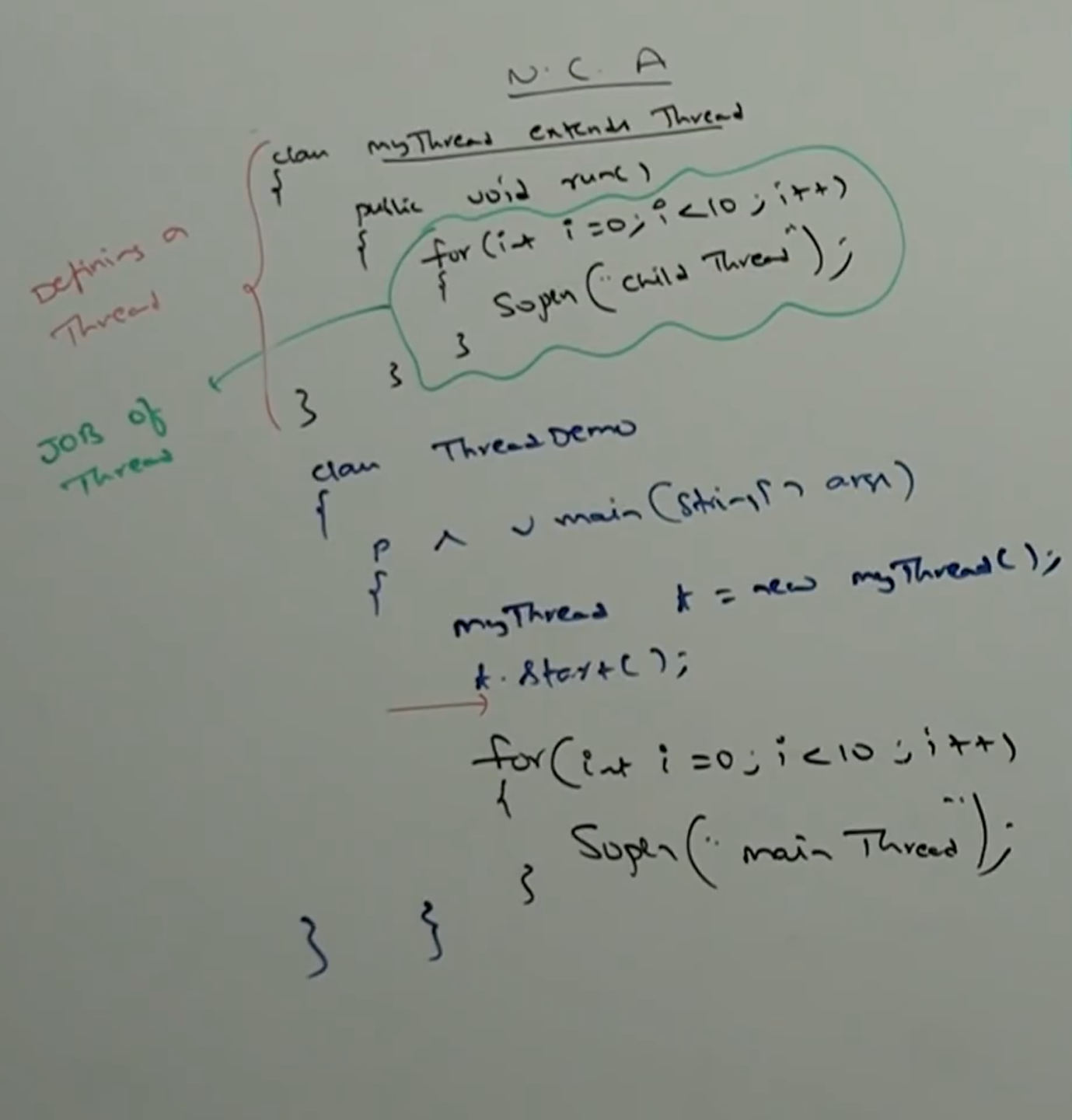
We are declaring a class that extends PopCorn without name (anonymous inner class).

In that child class we are overriding taste method.

For that child class we are creating an object with parent reference.

* **Defining a Thread by extending Thread Class:**

**Normal class approach:**

****

**Anonymous inner class approach:**

class ThreadDemo{

public static void main(String[] args){

Thread t = new Thread(){

public void run(){

for(int i = 0; i <10; i++){

System.out.println(“Child Thread”);

}

}

};

t1.start();

for(int i = 0; i <10; i++){

System.out.println(“Main Thread”);

}

}

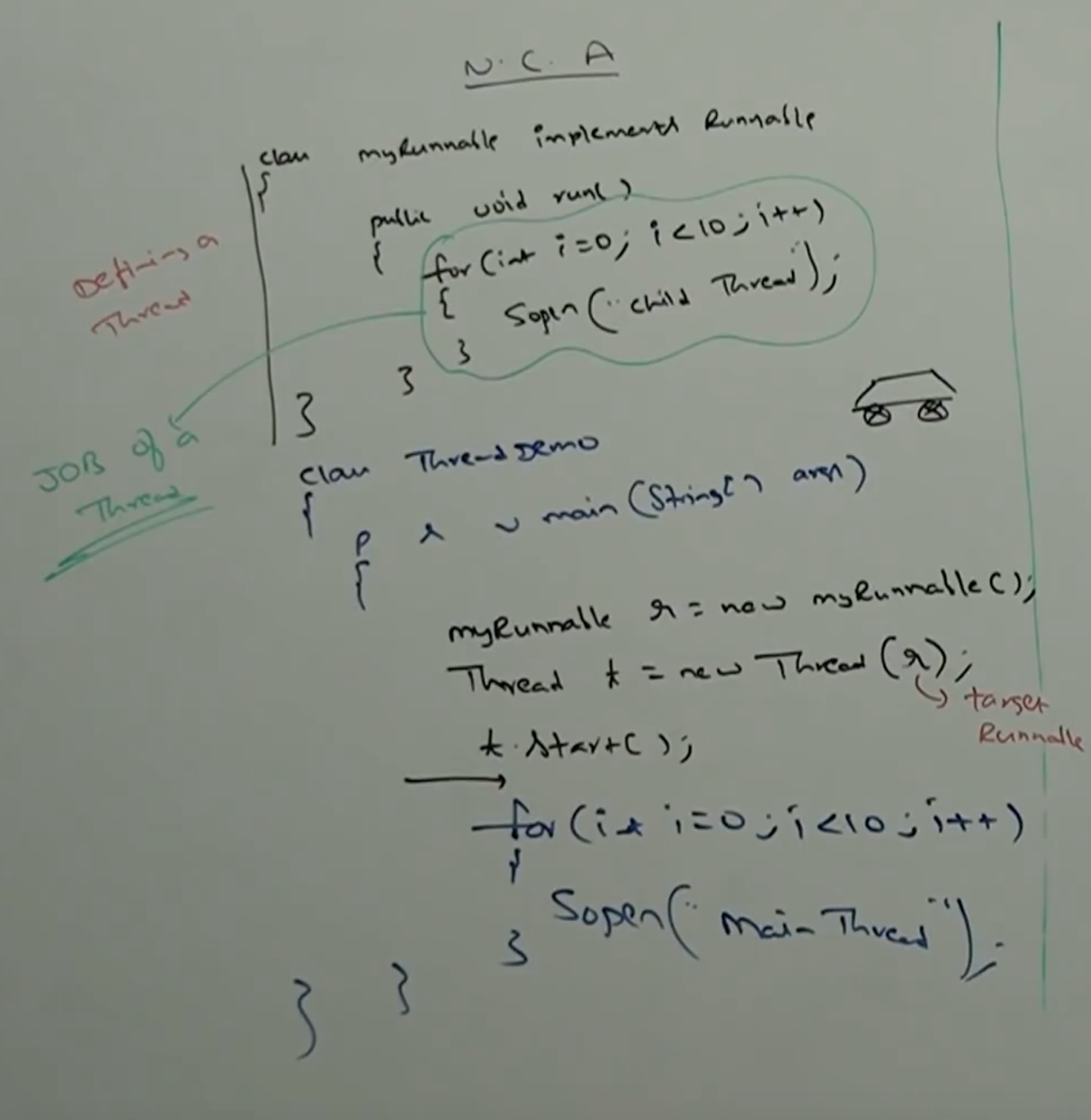
}

**Output:**

* **Anonymous Inner class that implements an Interface:**

**Defining a thread by implementing Runnable interface:**

Normal class approach:



Anonymous inner class approach:

class ThreadDemo{

public static void main(String[] args){

Runnable r = new Runnable(){

public void run(){

for(int i = 0; i < 10; i++){

System.out.println(“Child Thread-1”);

}

}

};

Thread t = new Thread(r);

t.start();

for(int i = 0; i < 10; i++){

System.out.println(“Child Thread-1”);

}

}

}

* **Anonymous inner class that defined inside arguments:**

class ThreadDemo{

public static void main(String[] args){

new Thread( new Runnable(){

public void run(){

for(int i = 0; i < 10; i++){

System.out.println(“ChildThread”);

}

}

}

).start();

}

for(int i = 0; i < 10; i++){

System.out.println(“Main Thread-1”);

}

}