Executor Service Runner-

ExecutorService executorService= Executors.*newSingleThreadExecutor*();

This mean it is a single thread can be executed at once,executing multiple tasks one by one.

executorService.execute(**new** Task1());

executorService.execute(**new** Thread(**new** Task2()));

This means second task will be executed only when first tasks is completely executed.

For example:This is task 1

**package** learnings;

**public** **class** Task1 **extends** Thread{

**public** **void** run() {

System.***out***.println("Thread 1 Started");

**for**(**int** i=0;i<=100;i++) {

System.***out***.print(" "+i);

}

System.***out***.println("Thread 1 Ended");

}

}

## This is task2:

**package** learnings;

**public** **class** Task2 **implements** Runnable {

@Override

**public** **void** run() {

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

System.***out***.println("Thread 2 Started");

**for**(**int** i=100;i<=199;i++) {

System.***out***.print(" "+i);

}

System.***out***.println("Thread 2 Ended");

}

}

## ExecutorService:

**public** **class** ExecutorServiceRunner {

**public** **static** **void** main(String[] args) {

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

ExecutorService executorService= Executors.*newSingleThreadExecutor*();

executorService.execute(**new** Task1());

executorService.execute(**new** Thread(**new** Task2()));

}

o/p:THREAD1->1…….100

Thread2-> 101……………..200

If you Directly execute some task directly within code it will be executed and will be executed parallel.

Thread 1 Started

**Task 3 kicked off**

0 300 301 302 303 304 305 306 307 308 309 310 311 312 1 2 313 3 314 4 315 5 316 6 317 7 8 318 319 320 321 322 323 324 9 325 326 327 328 10 11 12 13 329 330 331 332 333 334 335 336 14 15 16 17 18 19 20 21 22 23 24 25 337 338 339 340 341 26 27 28 29 30 31 32 33 34 35 36 37 38 39 342 343 344 345 346 347 40 348 349 350 351 352 353 354 41 355 356 357 358 359 42 43 44 45 46 360 361 362 363 364 365 366 367 368 369 47 48 49 50 51 52 53 370 371 372 373 374 54 55 375 376 377 378 379 380 56 381 382 383 384 385 386 57 58 59 60 61 62 387 388 389 390 391 392 393 394 63 64 65 66 395 67 396 68 69 70 71 72 73 74 75 76 77 78 397 79 398 80 399 81 400 82Task3 Done

Main Done

83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100Thread 1 Ended

Thread 2 Started

100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199Thread 2 Ended.

There are different ways to start a thread

For the class which extends Threads is used

Task1 obj1=new Task1();

Obj1.start();

For the class which implements Runnable

Task2 obj2=new Task2();

Thread task2Thread=new Thread(obj2);

task2Thread.start();

for Example-

**public** **static** **void** main(String[] args) {

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

Task1 Obj1=**new** Task1();

Obj1.start();

Task2 Obj2=**new** Task2();

Thread t=**new** Thread(Obj2);

t.start();

}