```
1. //
2. //this_java_11장_스레드_기본소스
3. //
4. //package sec02.exam01_createthread;
5. //
6. import java.awt.Toolkit;
7. public class BeepPrintExample1 {
8.         public static void main(String[] args) {
9.                 Toolkit toolkit = Toolkit.getDefaultToolkit();
10.                for(int i=0; i<5; i++) {
11.                        toolkit.beep();
12.                        System.out.println("띵");
13.                        try { Thread.sleep(500); } catch(Exception e) { }
14.                }
15.        }
16. }
17. //
18. public class BeepTask implements Runnable {
19.        public void run() {
20.                Toolkit toolkit = Toolkit.getDefaultToolkit();
21.                for(int i=0; i<5; i++) {
22.                        toolkit.beep();
23.                        System.out.println("띵");
24.                        try { Thread.sleep(500); } catch(Exception e) { }
25.                }
26.        }
27. }
28. //
29. public class BeepThread extends Thread {
30.        @Override
31.        public void run() {
32.                Toolkit toolkit = Toolkit.getDefaultToolkit();
33.                for(int i=0; i<5; i++) {
34.                        toolkit.beep();
35.                        System.out.println("띵");
36.                        try { Thread.sleep(500); } catch(Exception e) { }
37.                }
38.        }
39. }
40. //
41. public class BeepPrintExample2 {
42.        public static void main(String[] args) {
43.                //how1
44.                Runnable beepTask = new BeepTask();
45.                Thread thread = new Thread(beepTask);
46.
47.                //how2
48.                /*Thread thread = new Thread(new Runnable() {
49.                        @Override
50.                        public void run() {
51.                                Toolkit toolkit = Toolkit.getDefaultToolkit();
52.                                for(int i=0; i<5; i++) {
53.                                        toolkit.beep();
54.                                        System.out.println("띵");
55.                                        try { Thread.sleep(500); } catch(Exception e) { }
56.                                }
57.                        }
58.                });*/
59.
60.                //how3
61.                /*Thread thread = new Thread(() -> {
62.                        Toolkit toolkit = Toolkit.getDefaultToolkit();
63.                        for(int i=0; i<5; i++) {
64.                                toolkit.beep();
65.                                System.out.println("띵");
66.                                try { Thread.sleep(500); } catch(Exception e) { }
67.                        }
68.                });*/
```

```
69.
70.                 thread.start();
71.         }
72. }
73. //
74. public class BeepPrintExample3 {
75.         public static void main(String[] args) {
76.                 //how1
77.                 Thread thread = new BeepThread();
78.
79.                 //how2
80.                 /*Thread thread = new Thread() {
81.                         @Override
82.                         public void run() {
83.                                 Toolkit toolkit = Toolkit.getDefaultToolkit();
84.                                 for(int i=0; i<5; i++) {
85.                                         toolkit.beep();
86.                                         System.out.println("띵");
87.                                         try { Thread.sleep(500); } catch(Exception e) { }
88.                                 }
89.                         }
90.                 };*/
91.
92.                 thread.start();
93.         }
94. }
95. //
96. //package sec02.exam02_threadname;
97. //
98. public class ThreadA extends Thread {
99.         public ThreadA() {
100.                setName("ThreadA");
101.        }
102.        public void run() {
103.                for(int i=0; i<2; i++) {
104.                        System.out.println(getName() + "가 출력한 내용");
105.                }
106.        }
107. }
108. //
109. public class ThreadB extends Thread {
110.        public void run() {
111.                for(int i=0; i<2; i++) {
112.                        System.out.println(getName() + "가 출력한 내용");
113.                }
114.        }
115. }
116. //
117. public class ThreadNameExample {
118.        public static void main(String[] args) {
119.                Thread mainThread = Thread.currentThread();
120.                System.out.println("프로그램 시작 스레드 이름: " + mainThread.getName());
121.
122.                ThreadA threadA = new ThreadA();
123.                System.out.println("작업 스레드 이름: " + threadA.getName());
124.                threadA.start();
125.
126.                ThreadB threadB = new ThreadB();
127.                System.out.println("작업 스레드 이름: " + threadB.getName());
128.                threadB.start();
129.        }
130. }
131. //
132. //package sec03.exam01_priority;
133. //
134. public class CalcThread extends Thread {
135.        public CalcThread(String name) {
136.                setName(name);
```

```
137.     }
138.     public void run() {
139.             for(int i=0; i<2000000000; i++) { }
140.             System.out.println(getName());
141.     }
142. }
143. //
144. public class PriorityExample {
145.     public static void main(String[] args) {
146.             for(int i=1; i<=10; i++) {
147.                     Thread thread = new CalcThread("thread" + i);
148.
149.                     if(i != 10) {
150.                             thread.setPriority(Thread.MIN_PRIORITY);
151.                     } else {
152.                             thread.setPriority(Thread.MAX_PRIORITY);
153.                     }
154.                     thread.start();
155.             }
156.     }
157. }
158. //
159. //package sec04.exam01_unsynchronized;
160. //package sec04.exam02_synchronized;
161. //
162. //unsynchronized Calculator
163. //
164. public class Calculator {
165.     private int memory;
166.
167.     public int getMemory() {
168.             return memory;
169.     }
170.     public void setMemory(int memory) {
171.             this.memory = memory;
172.             try {
173.                     Thread.sleep(2000);
174.             } catch(InterruptedException e) { }
175.
176.             System.out.println(Thread.currentThread().getName() + ": " +  this.memory);
177.     }
178. }
179. //
180. //synchronized Calculator
181. //
182. public class Calculator {
183.     private int memory;
184.
185.     public int getMemory() {
186.             return memory;
187.     }
188.     public synchronized void setMemory(int memory) {
189.             this.memory = memory;
190.             try {
191.                     Thread.sleep(2000);
192.             } catch(InterruptedException e) { }
193.
194.             System.out.println(Thread.currentThread().getName() + ": " +  this.memory);
195.     }
196. }
197. //
198. public class User1 extends Thread {
199.     private Calculator calculator;
200.
201.     public void setCalculator(Calculator calculator) {
202.             this.setName("User1");
203.             this.calculator = calculator;
204.     }
```

```
205.     public void run() {
206.             calculator.setMemory(100);
207.     }
208. }
209. //
210. public class User2 extends Thread {
211.     private Calculator calculator;
212.
213.     public void setCalculator(Calculator calculator) {
214.             this.setName("User2");
215.             this.calculator = calculator;
216.     }
217.     public void run() {
218.             calculator.setMemory(50);
219.     }
220. }
221. //
222. public class MainThreadExample {
223.     public static void main(String[] args) {
224.             Calculator calculator = new Calculator();
225.
226.             User1 user1 = new User1();
227.             user1.setCalculator(calculator);
228.             user1.start();
229.
230.             User2 user2 = new User2();
231.             user2.setCalculator(calculator);
232.             user2.start();
233.     }
234. }
235. //
236. //package sec05.exam01_state;
237. //
238. public class StatePrintThread extends Thread {
239.     private Thread targetThread;
240.
241.     public StatePrintThread(Thread targetThread) {
242.             this.targetThread = targetThread;
243.     }
244.     public void run() {
245.             while(true) {
246.                     Thread.State state = targetThread.getState();
247.                     System.out.println("타겟 스레드 상태: " + state);
248.
249.                     if(state == Thread.State.NEW) {
250.                             targetThread.start();
251.                     }
252.                     if(state == Thread.State.TERMINATED) {
253.                             break;
254.                     }
255.                     try {
256.                             Thread.sleep(500); //0.5초간 일시 정지
257.                     } catch(Exception e) { }
258.             }
259.     }
260. }
261. //
262. public class TargetThread extends Thread {
263.     public void run() {
264.             for(long i=0; i<1000000000; i++) { }
265.
266.             try {
267.                     Thread.sleep(1500); //1.5초간 일시 정지
268.             } catch(Exception e) { }
269.
270.             for(long i=0; i<1000000000; i++) { }
271.     }
272. }
```

```
273. //
274. public class ThreadStateExample {
275.     public static void main(String[] args) {
276.             StatePrintThread statePrintThread = new StatePrintThread(new TargetThread());
277.             statePrintThread.start();
278.     }
279. }
280. //
281. //package sec06.exam01_sleep;
282. //
283. import java.awt.Toolkit;
284. public class SleepExample {
285.     public static void main(String[] args) {
286.             Toolkit toolkit = Toolkit.getDefaultToolkit();
287.
288.             for(int i=0; i<10; i++) {
289.                     toolkit.beep();
290.                     System.out.println("띵");
291.                     try {
292.                             Thread.sleep(3000);
293.                     } catch(InterruptedException e) { }
294.             }
295.     }
296. }
297. //
298. //package sec06.exam02_yield;
299. //
300. public class ThreadA extends Thread {
301.     public boolean stop = false;
302.     public boolean work = true;
303.
304.     public void run() {
305.             while(!stop) {
306.                     if(work) { System.out.println("ThreadA 작업 내용"); }
307.                     else { Thread.yield(); }
308.             }
309.             System.out.println("ThreadA 종료");
310.     }
311. }
312. //
313. public class ThreadB extends Thread {
314.     public boolean stop = false;
315.     public boolean work = true;
316.
317.     public void run() {
318.             while(!stop) {
319.                     if(work) { System.out.println("ThreadB 작업 내용"); }
320.                     else { Thread.yield(); }
321.             }
322.             System.out.println("ThreadB 종료");
323.     }
324. }
325. //
326. public class YieldExample {
327.     public static void main(String[] args) {
328.             ThreadA threadA = new ThreadA();
329.             ThreadB threadB = new ThreadB();
330.
331.             threadA.start();
332.             threadB.start();
333.
334.             try { Thread.sleep(3000); } catch (InterruptedException e) {}
335.             threadA.work = false;
336.
337.             try { Thread.sleep(3000); } catch (InterruptedException e) {}
338.             threadA.work = true;
339.
340.             try { Thread.sleep(3000); } catch (InterruptedException e) {}
```

```
341.                 threadA.stop = true;
342.                 threadB.stop = true;
343.     }
344. }
345. //
346. //package sec06.exam03_join;
347. //
348. public class SumThread extends Thread {
349.     private long sum;
350.
351.     public long getSum() { return sum; }
352.     public void setSum(long sum) { this.sum = sum; }
353.
354.     public void run() {
355.                 for(int i=1; i<=100; i++) { sum+=i; }
356.     }
357. }
358. //
359. public class JoinExample {
360.     public static void main(String[] args) {
361.                 SumThread sumThread = new SumThread();
362.                 sumThread.start();
363.                 try {
364.                             sumThread.join();
365.                 } catch (InterruptedException e) {
366.                 }
367.                 System.out.println("1~100 합: " + sumThread.getSum());
368.     }
369. }
370. //
371. //sec06.exam04_wait_notify;
372. //
373. public class ThreadA extends Thread {
374.     private WorkObject workObject;
375.
376.     public ThreadA(WorkObject workObject) {
377.                 this.workObject = workObject;
378.     }
379.     @Override
380.     public void run() {
381.                 for(int i=0; i<10; i++) { workObject.methodA(); }
382.     }
383. }
384. //
385. public class ThreadB extends Thread {
386.     private WorkObject workObject;
387.
388.     public ThreadB(WorkObject workObject) {
389.                 this.workObject = workObject;
390.     }
391.     @Override
392.     public void run() {
393.                 for(int i=0; i<10; i++) { workObject.methodB(); }
394.     }
395. }
396. //
397. public class WorkObject {
398.     public synchronized void methodA() {
399.                 System.out.println("ThreadA의 methodA() 작업 실행");
400.                 notify();
401.
402.                 try { wait(); } catch (InterruptedException e) { }
403.     }
404.     public synchronized void methodB() {
405.                 System.out.println("ThreadB의 methodB() 작업 실행");
406.                 notify();
407.
408.                 try { wait(); } catch (InterruptedException e) { }
```

```
409.     }
410. }
411. //
412. public class WaitNotifyExample {
413.     public static void main(String[] args) {
414.             WorkObject sharedObject = new WorkObject();
415.
416.             ThreadA threadA = new ThreadA(sharedObject);
417.             ThreadB threadB = new ThreadB(sharedObject);
418.             threadA.start();
419.             threadB.start();
420.     }
421. }
422. //
423. //package sec06.exam05_wait_notify;
424. //
425. public class DataBox {
426.     private String data;
427.
428.     public synchronized String getData() {
429.             if(this.data == null) {
430.                     try { wait(); } catch(InterruptedException e) {}
431.             }
432.             String returnValue = data;
433.             System.out.println("ConsummerThread가 읽은 데이터: " + returnValue);
434.             data = null;
435.             notify();
436.
437.             return returnValue;
438.     }
439.     public synchronized void setData(String data) {
440.             if(this.data != null) {
441.                     try { wait(); } catch(InterruptedException e) {}
442.             }
443.             this.data = data;
444.             System.out.println("ProducerThread가 생성한 데이터: " + data);
445.             notify();
446.     }
447. }
448. //
449. public class ProducerThread extends Thread {
450.     private DataBox dataBox;
451.
452.     public ProducerThread(DataBox dataBox) { this.dataBox = dataBox; }
453.
454.     @Override
455.     public void run() {
456.             for(int i=1; i<=3; i++) {
457.                     String data = "Data-" + i;
458.                     dataBox.setData(data);
459.             }
460.     }
461. }
462. //
463. public class ConsumerThread extends Thread {
464.     private DataBox dataBox;
465.
466.     public ConsumerThread(DataBox dataBox) { this.dataBox = dataBox; }
467.
468.     @Override
469.     public void run() {
470.             for(int i=1; i<=3; i++) {
471.                     String data = dataBox.getData();
472.             }
473.     }
474. }
475. //
476. public class WaitNotifyExample {
```

```
477.     public static void main(String[] args) {
478.             DataBox dataBox = new DataBox();
479.
480.             ProducerThread producerThread = new ProducerThread(dataBox);
481.             ConsumerThread consumerThread = new ConsumerThread(dataBox);
482.             producerThread.start();
483.             consumerThread.start();
484.     }
485. }
486. //
487. //package sec06.exam06_stop;
488. //
489. public class PrintThread1 extends Thread {
490.     private boolean stop;
491.
492.     public void setStop(boolean stop) { this.stop = stop; }
493.
494.     public void run() {
495.             while(!stop) { System.out.println("실행 중"); }
496.             System.out.println("자원 정리");
497.             System.out.println("실행 종료");
498.     }
499. }
500. //
501. public class StopFlagExample {
502.     public static void main(String[] args)  {
503.             PrintThread1 printThread = new PrintThread1();
504.             printThread.start();
505.
506.             try { Thread.sleep(1000); } catch (InterruptedException e) { }
507.
508.             printThread.setStop(true);
509.     }
510. }
511. //
512. public class InterruptExample {
513.     public static void main(String[] args)  {
514.             Thread thread = new PrintThread2();
515.             thread.start();
516.
517.             try { Thread.sleep(1000); } catch (InterruptedException e) { }
518.
519.             thread.interrupt();
520.     }
521. }
522. //
523. public class PrintThread2 extends Thread {
524.     public void run() {
525.             //how1
526.             /*try {
527.                     while(true) {
528.                             System.out.println("실행 중");
529.                             Thread.sleep(1);
530.                     }
531.             } catch(InterruptedException e) {
532.             }*/
533.
534.             //how2
535.             while(true) {
536.                     System.out.println("실행 중");
537.                     if(Thread.interrupted()) { break; }
538.             }
539.             System.out.println("자원 정리");
540.             System.out.println("실행 종료");
541.     }
542. }
```