به نام خدا

سیستم عامل

استاد: دکتر خانمیرزا

سینکرونیزیشن و ددلاک

تمرین

ترم 992

سوالات کاغذی

1. توابع wait و signal سمافور busyWaiting را با استفاده از دستور testAndSet بنویسید؟ (5 نمره)
2. در سیستمی **4** پردازش و **5** منبع وجود دارد. هر منبع فقط یک نمونه دارد. اگر وضعیت جاری سیستم به صورت زیر باشد، کدام گزینه درست است؟ (راه حل خود را توضیح دهید) (5 نمره)

نیاز (Current Need) تخصیص (Current Allocation)

	r1	r2	r3	r4	r5	r1	r2	r3	r4	r5
p1	0	0	0	1	0	1	0	0	0	0
p2	1	0	0	0	1	0	1	0	0	0
p3	0	1	0	0	0	0	0	1	0	0
p4	0	0	1	0	0	0	0	0	1	1

1) سیستم در حالت امن است. 2) سیستم در حالت بن بست است

3) ممکن است سیستم به حالت ناامن برود 4)ممکن است سیستم به بن بست برود.

3. در یک سامانه نرم افزاری کد زیر نوشته شده و برای مدیریت منبع مشترک از قفل (لاک) استفاده شده است. یکبار به کمک سمافور و یکبار با استفاده از مونیتور (کلمه کلیدی synchronized) آن را باز نویسی کنید؟(6)

```java
public class TimeSlotService {
    private final TimeSlotRepository timeSlotRepo;
    private final Lock lock = new ReentrantLock();

    public void chooseSlot(String slotId) {
        lock.lock();
        Optional<TimeSlot> timeSlot = timeSlotRepo.findById(slotId);
        if (timeSlot.isEmpty() || timeSlot.get().getStudentId() != null) {
            lock.unlock();
            return;
        }
        String sid = SecurityUtil.getCurrentUser();
        TimeSlot slot = timeSlot.get();
        slot.setStudentId(sid);
        timeSlotRepo.save(slot);
        lock.unlock();
    }
```

سوالات پیاده سازی. (با زبان جاوا) (هر مورد 14 نمره)

4. دو کلاس زیر را در نظر بگیرید (Resource و ResourcePool) : (از روی متن کپی کنید!)

```
public class Resource {
    private static long idGenerator = 1L;
    private final Long id;

    public Resource() {
        this.id = idGenerator++;
    }

    public void useResource() {
        final String uid = UUID.randomUUID().toString();
        final String str = ThreadLocalRandom
                .current().nextBoolean() ? "####" : ThreadLocalRandom
                .current().nextBoolean() ? "$$$$$$" : "@";
        System.out.println(str + " Using Resource with ID= "
                + id + " Started at " + System.nanoTime()
                + " Call ID = " + uid + " " +  str);
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println(str + " Using Resource with ID= "
                + id + " Ended at " + System.nanoTime()
                + " Call ID = " + uid + " " + str);
    }
}
```

```
public class ResourcePool {
    // TODO

    public ResourcePool(int initialSize) {
        // TODO
    }

    public void useResource() {
        // TODO
    }
}
```

میخواهیم کلاس ResourcePool را به گونه ای بنویسیم، که در ابتدا هنگام ساخت، به تعداد ورودی سازنده، Resource ساخته و نگه داریم، هنگام فراخوانی متد useResource یکی از Resource ها انتخاب شده و متد useResource آن فراخوانی شود.

اما میخواهیم به گونه ای بنویسیم، که هنگام فراخوانی useResource، Resource مورد استفاده برای ریسمان های دیگر قابل استفاده نباشد و در هر لحظه تنها یک ریسمان، در حال استفاده از یک Resource باشد.

همچنین در صورتی که Resource ای برای انتخاب و صدا زدن متد مورد نظر در حال حاضر موجود نبود، باید Resource جدیدی ساخته شود و در ResourcePool نگه داشته شود. (برای پیاده سازی از لاک و یا سمافور استفاده کنید)

توجه کنید که کلاس Resource را نباید عوض کنید، تنها کلاس ResourcePool باید عوض شود.

برای تست اجرا از کد زیر میتوانید استفاده کنید:

```
public static void main(String[] args) {
    ResourcePool pool = new ResourcePool(10);
    IntStream.range(0,100)
            .parallel()
            .peek(System.out::println)
            .forEach(z-> new Thread(()-> pool.useResource()).start());
}
```

---

5. دو کلاس Player و Server را در نظر بگیرید: (کپی کنید)

```
public class Player extends Thread {
    private final String name;
    private final Server server;

    public Player(String name, Server server) {
        this.name = name;
        this.server = server;
    }
    @Override
    public void run(){
        System.out.println("player {" + name + "} is trying to join server!");
        int id = server.join(this);
        try {
            Thread.sleep(ThreadLocalRandom.current().nextLong(2000,10000));
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        server.ready(id);
        System.out.println("|| player with name " + name + " is ready!");
    }

    public void startGame(){
        System.out.println("Player " + name + "startedGame");
    }
}
```

```
public class Server implements Runnable{
    private int players;

    public Server(int lobbySize){
        players = lobbySize;
    }
    @Override public void run() {
        // TODO
    }
    public int join(Player player){
        int id = 0;
        // TODO

        System.out.println("-> player {" + player.getName() + "} is joining server!");
        return 10; // FIXME
    }
    public void ready(int id){
        // TODO
    }
}
```

کلاس سرور و پلیر داریم، در ابتدا مشخص می شود (در سازنده کلاس سرور) که چه تعداد پلیر باید به سرور متصل شوند و آمادگی خود را برای شروع از طریق متد ready اعلام کنند، روند به این شکل است که ابتدا هر پلیر متد join از سرور را صدا می زند و به سرور می پیوندد و هنگامی که آماده شد، متد ready را فراخوانی کرده و id اختصاص داده شده به خود از سمت سرور را نیز پاس می دهد تا سرور بفهمد که کدام پلیر آماده است.

می خواهیم سرور را طوری پیاده کنیم، که هنگامی که به تعداد داده شده پلیر به سرور اضافه شده بود، و همه اعلام آمادگی کردند، سرور متد startGame تک تک پلیر ها را فراخوانی کند. برای پیاده سازی از سمافور و یا لاک استفاده کنید. (استفاده از ساختار های پیشرفته تر همچون Phaser و CountDownLatch و CyclicBarrier و .. مجاز نیست)

برای تست اجرا از کد زیر میتوانید استفاده کنید:

```
public static void main(String[] args) {
    final int SIZE = 15;
    Server server = new Server(SIZE);
    new Thread(server).start();
    IntStream.range(0, SIZE)
            .parallel()
            .forEach(z -> new Player(UUID.randomUUID().toString(), server)
                    .start());
}
```

---

6. کلاس Printer با متد print(String text) که محتوای گرفته شده را چاپ میکند را در نظر بگیرید. با فرض اینکه پرینتر در هر لحظه به طور همزمان می تواند N محتوا را چاپ کند، پرینتر را پیاده سازی کنید. (دقت کنید که مدیریت اینکه در هر لحظه بیشتر از N تا ترد، چیزی چاپ نکنند (بلاک شوند، تا کار یکی تمام شوند) باید در همین کلاس مدیریت شود. (پیاده سازی یکبار با قفل و یکبار دیگر با سمافور)

---

7. یک پل 4 بانده دو طرفه به طول یک وسیله نقلیه باید را در نظر بگیرید که برای وارد شدن و خارج شدن از یک شهر استفاده می شود، و همچنین دو نوع وسیله نقلیه خودرو معمولی به عرض 1 و کامیون به عرض 4 را در نظر بگیرید. برنامه ای پیاده کنید که چنین ساختاری را پیاده کند و تعدادی وسیله نقلیه را از پل عبور دهد. (واضح است در بهترین حالت 4 وسیله نقلیه (4 خودرو معمولی) و در بدترین حالت 1 وسیله نقلیه (کامیون) می تواند روی پل باشد.) این برنامه را یکبار با سمافور و یکبار با قفل (مونیتور) پیاده کنید.

واضح است که هر وسیله نقلیه یک ترد است و مدیریت اینکه درخواست وسیله نقلیه پاسخ داده شود یا بلاک شود و بعدا اجرا شود در شی Bridge مدیریت می شود و وسایل نقلیه آگاهی خاصی از پل ندارند و تنها یکی از دو متد وارد یا خارج شدن را فراخوانی می کنند.

---

8. سوال 6 را اینبار به گونه ای پیاده کنید، که الویت کامیون بیشتر از خودرو باشد، به طوریکه اگر کامیونی منتظر بود، اجازه ورود هیچ گونه خودرویی داده نشود تا کامیون عبور کند.(سمافور یا قفل - نیاز به هر دو پیاده سازی نیست)

---

9. سوال 6 را اینبار به گونه ای پیاده کنید که طول پل N باشد به این معنی که روی هر لاین هر لحظه می تواند N خودرو وجود داشته باشد. (یکبار با سمافور و یکبار با استفاده از قفل)

به صورت یک فایل ZIP شامل کد ها، و یک فایل pdf برای 3 سوال اول آپلود کنید.

موفق باشید!