# BSCCS2005: Practice with Solutions
Week 10

Two passengers $P_1$ and $P_2$ are trying to book sleeper class tickets on the same train. Suppose there is exactly one sleeper ticket left on that train and that there is no provision to book RAC and/or waiting list tickets. Consider the Java code modeling the given scenario, and answer questions 1 and 2.

---

```
1  class Passenger implements Runnable{
2      private String name;
3      private int coach_no, seat_no;
4      public boolean booked = false;
5      public Passenger(String pass_name) {
6          name = pass_name;
7      }
8      public void run() {
9          coach_no = 10;
10         seat_no = 52;
11         booked = true;
12     }
13     public String getSeatAndPNR() {
14         return (coach_no+" "+seat_no);
15     }
16     public String getName() {
17         return name;
18     }
19 }
20 public class RailwayBooking {
21         public static void main(String args[]) {
22                 Passenger p1 = new Passenger("P1");
23                 Passenger p2 = new Passenger("P2");
24                 Thread t1 = new Thread(p1);
25                 Thread t2 = new Thread(p2);
26                 if (p1.booked == false && p2.booked == false) {
27                         t1.start();
28                 }
29                 if (p1.booked == false && p2.booked == false) {
30                         t2.start();
31                 }
32                 System.out.println(p1.getName()+" "+p1.getSeatAndPNR());
33                 System.out.println(p2.getName()+" "+p2.getSeatAndPNR());
34         }
35 }
```

1. Identify the critical section in this code.

   √ Lines 9 to 11

   ◯ Lines 26 to 31

   ◯ Lines 32 to 33

   ◯ There is no critical section in this code.

**Solution:** The same resource, that is seat number 52 in coach number 10, may be booked by both $P_1$ and $P_2$ due to lines in 10 to 12 being executed by objects of both passengers.

2. Which of the following is/are NOT possible outcome(s) of the execution of the given code?

   ◯
   ```
   P1 0 0
   P2 10 52
   ```

   √
   ```
   P2 10 52
   P1 0 0
   ```

   ◯
   ```
   P1 10 52
   P2 10 52
   ```

   ◯
   ```
   P1 10 0
   P2 10 52
   ```

**Solution:** Option 1 is possible because the main program is also running as a thread. Hence, the main program may execute line 32 before t1 executes line 10 and 11 (before setting the coach_no and seat_no of p1).

Order of execution:
32, 10, 11

Option 3 is possible when both t1 and t2 are executed before main executes line 33.

Order of execution:
9 - 11 for both t1 and t2, then 33

Option 4 is possible when both t1 and t2 are executed before main executes line 32.

Order of execution:
Line 9 of t1 (thread t1)
Line 9 to 11 of t2 (thread t2)
Line 32 (main thread)
Line 14 of t1 (thread t2)
Line 10 of t1 (thread t1)

In whatever the order the threads may get executed, the values of p1 will be printed first. Hence, option 2 will never be the correct output.

3. Consider the Java code given below.

```
class Expression implements Runnable{
    static int a = 1, b = 2;
    static boolean ready_1=false, ready_2=true;
    public void run() {
    }
}
class Expression1 extends Expression {
    public void run() {
        while (Expression.ready_2) {}
        Expression.a = 3 * (Expression.b/2);
        Expression.a = Expression.a * Expression.b;
        Expression.ready_1 = false;
    }
}
class Expression2 extends Expression{
    public void run() {
        while (Expression.ready_1) {}
        Expression.a = 4 * (Expression.b/2);
        Expression.a = Expression.a * Expression.b;
        Expression.ready_2 = false;
    }
}
public class ExpressionEval {
    public static void main(String[] args) {
        Expression expr1 = new Expression1();
        Expression expr2 = new Expression2();
        Thread t1 = new Thread(expr1);
        Thread t2 = new Thread(expr2);
        t1.start();
        t2.start();
        Thread.sleep(10); //try catch removed to simplify the code
    }
}
```

Assuming that the program terminates normally after a certain amount of time, which among the following is/are the possible value/s of `Expression.a` after the execution is completed?

- √ 6
- ◯ 8
- ◯ 12
- ◯ 16

**Solution:** The codes inside both the `run()` methods are mutually excluded from being interleaved by using the two variables `ready_1` and `ready_2`. Hence, when the code finishes execution, the value computed will be the value obtained by executing run() of Expression 2, followed by executing run() of Expression 1.

4. Consider the code given below.

```
class PrlCls extends Thread{
    private int init;
    public PrlCls(int n){
        init = n;
    }
    public void run(){
        for(int i = init; i <= init + 5; i++){
            System.out.print(i + " ");
            try{
                Thread.sleep(1000);
            }
            catch(InterruptedException e){}
        }
    }
}
public class FClass{
    public static void main(String[] args){
        Thread t1 = new PrlCls(10);
        Thread t2 = new PrlCls(20);
        t1.start();
        for(int i = 30; i < 35; i++){
            System.out.print(i + " ");
        }
        t2.start();
    }
}
```

Choose the correct option regarding the code.

- ◯ It may print 10 to 15, 20 to 25 and 30 to 35 in an interleaved manner.
- ◯ It may print 10 to 15 and 30 to 35 in an interleaved manner. However, 20 to 25 will always be printed after 10 to 15 and 30 to 35 are printed.
- ◯ It may print 10 to 15 and 30 to 35 in an interleaved manner. It may print 30 to 35 and 20 to 25 in an interleaved manner. However, 10 to 15 and 20 to 25 cannot be printed in an interleaved manner.
- √ It may print 10 to 15 and 30 to 35 in an interleaved manner. It may print 10 to 15 and 20 to 25 in an interleaved manner. However, 30 to 35 and 20 to 25 cannot be printed in an interleaved manner.

**Solution:** Since, the thread `t2` starts after the values 30 to 35 are printed, 30 to 35 and 20 to 25 cannot be printed in an interleaved manner.

5. Consider there are three threads T1, T2, T3 and two shared variables named `a, b` with initial values 1 and 9 respectively in a program.
   Thread T1 will execute a pseudo code segment to assign values to variable `a`
   Thread T2 will execute a pseudo code segment to assign values to variable `b`
   Thread T3 will print "`Sum of a and b is 10`" if `a + b` = 10.

   **T1's code:**

```
forLoop(i=2; i<=5 ;i++){
    P(S3)
    P(S2)
    a = i
    V(S1)
    // LINE 1
}
```

   **T2's code:**

```
forLoop(j=8; j>=5 ; j--){
    P(S3)
    P(S1)
    b = j
    V(S2)
    // LINE 2
}
```

   **T3's code:**

```
k = 5
whileLoop(k>0){
     P(S1)
     P(S2)
     if (a + b == 10)
     Then print "Sum of a and b is 10"
     k--
     V(S3)
     // LINE 3
}
```

   **Assumptions:**

   1. A semaphore can have values ranging from 0 to 2
   2. If a semaphore has value 0 then P() operation can't be used on it.
   3. If a semaphore has value 2 then V() operation can't be used on it.

If this program prints the statement `Sum of a and b is 10` total 5 times on every execution. Then choose the correct set of instructions which should be given in place of LINE 1, 2, 3.

- √ `LINE 1`: V(S1), `LINE 2`: V(S2), `LINE3`: V(S3)
- ◯ `LINE 1`: V(S3), `LINE 2`: V(S2), `LINE 3`: V(S1)
- ◯ `LINE 1`: V(S2), `LINE 2`: V(S3), `LINE 3`: V(S1)
- ◯ `LINE 1`: V(S1), `LINE 2`: P(S2), `LINE 3`: V(S3)

**Solution:**
We would need at least 3 semaphores for this example.
The code blocks would be as follows.
**T1's code:**

```
forLoop(i=2; i<=5 ;i++){
    P(S3)
    P(S2)
    a = i
    V(S1)
    V(S1)
}
```

**T2's code:**

```
forLoop(j=8; j>=5 ; j--){
    P(S3)
    P(S1)
    b = j
    V(S2)
    V(S2)
}
```

**T3's code:**

```
k = 5
whileLoop(k>0){
    P(S1)
    P(S2)
    if (a + b == 10)
    Then print "Sum of a and b is 10"
    k--
```

```
    V(S3)
    V(S3)
}
```

We will initialize the semaphores as follows
S1 = 2, S2 = 2, S3 = 0. Will be discussed in detail in live session.

6. Consider the following code.

```
class Point implements Runnable{
    static int x, y;
    private String name;
    public Point(String n){
      this.name = n;
    }
    public void updatePoint(int a, int b){
      x = a;
      y = b;
    }
    public void display(){
      System.out.println("("+ x +","+ y + ")");
    }
    public void run(){
      if(name.equals("update")){
        updatePoint(this.x + 5, this.y + 6);
      }
      if(name.equals("display")){
        display();
      }
    }
}
public class PointUser {
    public static void main(String[] args){
        Point p1 = new Point("update");
        Point p2 = new Point("display");
        Thread t1 = new Thread(p1);
        Thread t2 = new Thread(p2);
        t2.start();
        t1.start();
    }
}
```

Which among the following is/are possible scenarios of execution (assuming order of execution within a thread is sequential)?

- √ `updatePoint()` of `t1` after `display()` of `t2`
- √ `display()` of `t1` after `updatePoint()` of `t2`
- ◯ `updatePoint()` of `t1` after `display()` of `t1`
- ◯ `updatePoint()` of `t2` after `display()` of `t2`

Consider the code given below, and answer questions 7 and 8.

```
class Bank{
    int balance = 100;
    void extract(String name,int withdrawal){
        if (balance>=withdrawal){
            System.out.println(name + " has withdrawn "+ withdrawal);
            balance = balance - withdrawal;
        }
        else {
            System.out.println(name+ ", cannot withdraw "+ withdrawal);
            System.out.println("Because current balance is " + balance);
        }
    }
}
class Withdrawal implements Runnable{
    Bank bk;
    String name;
    int money;
    Withdrawal(Bank ob, String name, int money){
        this.bk=ob;
        this.name=name;
        this.money=money;
    }
    public void run(){
        bk.extract(name, money);
    }
}
public class ThreadBank{
    public static void main(String[] args){
        Bank obj = new Bank();
        Thread t1=new Thread(new Withdrawal(obj, "Sun", 20));
        Thread t2=new Thread(new Withdrawal(obj, "Moon", 40));
        Thread t3=new Thread(new Withdrawal(obj, "Earth", 80));
        t1.start();
        t2.start();
        t3.start();
    }
}
```

7. Which among the following is/are possible output/s of the given program?

√
```
Earth has withdrawn 80
Sun has withdrawn 20
Moon has withdrawn 40
```

√
```
Sun has withdrawn 20
Moon has withdrawn 40
Earth cannot withdraw 80
Because current balance is 40
```

√
```
Earth has withdrawn 80
Moon cannot withdraw 40
Because current balance is 20
Sun has withdrawn 20
```

√
```
Moon has withdrawn 40
Sun has withdrawn 20
Earth has withdrawn 80
```

8. In the given code, suppose you change the statements

```
t1.start();
t2.start();
t3.start();
```

to

```
t1.run();
t2.run();
t3.run();
```

Then, which among the following is/are the possible output/s?

◯
```
Earth has withdrawn 80
Sun has withdrawn 20
Moon has withdrawn 40
```

√
```
Sun has withdrawn 20
Moon has withdrawn 40
Earth cannot withdraw 80
Because current balance is 40
```

◯
```
Earth has withdrawn 80
Moon cannot withdraw 40
Because current balance is 20
Sun has withdrawn 20
```

◯
```
Moon has withdrawn 40
Sun has withdrawn 20
Earth has withdrawn 80
```