MODULE 8q - Threaded Code

SHARED CODE

In the following program, the method get() in class SharedObject is shared by two run() methods, one in ThreadA and the other in ThreadB:

```
public class ThreadExA
 { public static void main(String[] args)
    { SharedObject pot = new SharedObject();
      ThreadA tA = new ThreadA(pot);
      ThreadB tB = new ThreadB(pot);
      tA.start();
      tB.start();
    }
 }

class SharedObject
 { public void get(String s, long n)
    { for (int i=0; i<3; i++)
       { System.out.printf("Got by %s%n", s);
         try
          { Thread.sleep(n);                 // Pause for a moment
          }
         catch(InterruptedException e)
          {}
       }
    }
 }

class ThreadA extends Thread
 { private SharedObject shob;

   public ThreadA(SharedObject ob)
    { this.shob = ob;
    }

   public void run()
    { for (int i=0; i<2; i++)
       { this.shob.get("A", 1000L);
         try
          { this.sleep(2000L);
          }
         catch(InterruptedException e)
          {}
       }
    }
 }

class ThreadB extends Thread
 { private SharedObject shob;

   public ThreadB(SharedObject ob)
    { this.shob = ob;
    }
   public void run()
    { for (int i=0; i<2; i++)
       { this.shob.get("B", 100L);
         try
          { this.sleep(2000L);
          }
         catch(InterruptedException e)
          {}
       }
    }
 }

// This yields:
//
// Got by A
// Got by B
// Got by B
// Got by B
// Got by A
// Got by A
// Got by B
// Got by B
// Got by B
// Got by A
// Got by A
// Got by A
```

TRY IT OUT

Key the above program in, compile it and run it. Are the results
as shown? If so, explain what is happening. If not, why not?

THE synchronized MODIFIER

Next add the synchronized modifier (note the spelling) to the
heading of the get() method:

```
public class ThreadExB
 { public static void main(String[] args)
    { SharedObject pot = new SharedObject();
      ThreadA tA = new ThreadA(pot);
      ThreadB tB = new ThreadB(pot);
      tA.start();
      tB.start();
    }
 }

class SharedObject
 { public synchronized void get(String s, long n)  // modified heading
    { for (int i=0; i<3; i++)
       { System.out.printf("Got by %s%n", s);
         try
          { Thread.sleep(n);
          }
         catch(InterruptedException e)
          {}
       }
    }
 }

class ThreadA extends Thread
 { private SharedObject shob;

   public ThreadA(SharedObject ob)
    { this.shob = ob;
    }

   public void run()
    { for (int i=0; i<2; i++)
       { this.shob.get("A", 1000L);
         try
          { this.sleep(2000L);
          }
         catch(InterruptedException e)
          {}
       }
    }
 }

class ThreadB extends Thread
 { private SharedObject shob;

   public ThreadB(SharedObject ob)
    { this.shob = ob;
    }

   public void run()
    { for (int i=0; i<2; i++)
       { this.shob.get("B", 100L);
         try
          { this.sleep(2000L);
          }
         catch(InterruptedException e)
          {}
       }
    }
 }

// This yields:
//
// Got by A
// Got by A
// Got by A
// Got by B
// Got by B
// Got by B
// Got by A
// Got by A
// Got by A
// Got by B
// Got by B
// Got by B
```

TRY IT OUT

Compile and run this version. Note that introducing the synchronized modifier means that only one thread at a time can access the get() method.