CS520 Week 6 Assignment

General Rules for Homework Assignments

- You are strongly encouraged to add comments throughout the program. Doing so will help your facilitator to understand your programming logic and grade you more accurately.
- You must work on your assignments individually. You are not allowed to copy the answers from the others. *However*, you are encouraged to discuss the approaches to the homework assignments with your section mates and the facilitator in your section via the discussion board.
- Each assignment has a strict deadline. However, you are still allowed to submit your assignment within 2 days after the deadline with a penalty. 15% of the credit will be deducted unless you made previous arrangements with your facilitator and professor. Assignments submitted 2 days after the deadline will not be graded.
- When the term *lastName* is referenced in an assignment, please replace it with your last name.

You are strongly encouraged to add comments into your program!

Create a new Java Project in Eclipse named HW6_*lastName* and complete the following requirements based on the Threads. Several threads will share a single object and contribute their individual result to the shared object. The shared object accumulates the partial results.

Part1 (50 points)

Create a package named *cs520.hw6.part1.* Using this package, create the following classes.

1. Create a class named *SharedResults* as follows. The class keeps track of the shared *results*.

    a. The instance (or member) private variable – *results* (ArrayList of integers).
    b. A default constructor that initializes the above data structure.
    c. A void *addToResults* method which takes the given integer argument and adds it to the shared *results*. This method then prints to the console the name of the current thread, the value it added, and the shared results data structure. Handle the synchronization issue with this method.
    d. The *getResult* method with no arguments which returns the sum of the values in the shared *results* data structure. Handle the synchronization issue with this method.

2. Create a class named *LongTask* which extends the *Thread* class.

    a. The instance (or member) private variables – *sharedData* (of type *SharedResults*), *start* (integer) and *end* (integer).
    b. A single constructor which takes the above three arguments and stores them in the instance values. Also, create a name for this thread as *Thread_<start>_<end>*
    c. In the *run* method, add the integer numbers from *start* to *end* (both inclusive) using a *for* loop. Also, sleep for a random time (up to 10 milliseconds) in each iteration of the loop. After the loop, invoke the *addToResults* method of the shared object and provide this accumulated sum.

3. Create a *Test* class to test the following functionality in its *main* method.
    a. Create the *SharedResults* object and assign it to a variable.
    b. Create five *LongTask* objects by passing the above shared object and the *start* and *end* values for each as (1, 100), (101, 200), (201, 300), (301, 400), and (401, 500) respectively.
    c. Start each thread as it is created.
    d. Wait for all the threads to complete using the *join* method.
    e. Print the *result* from the shared object.

Sample Output:

Different runs of the program will produce the output in different sequences, but the final result would be the same. Two runs of the program are shown below.

```
Thread_401_500 is adding 45050,
  Cumulative Results are [45050]
Thread_301_400 is adding 35050,
  Cumulative Results are [45050, 35050]
Thread_101_200 is adding 15050,
  Cumulative Results are [45050, 35050, 15050]
Thread_1_100 is adding 5050,
  Cumulative Results are [45050, 35050, 15050, 5050]
Thread_201_300 is adding 25050,
  Cumulative Results are [45050, 35050, 15050, 5050, 25050]
Final Result = 125250
```

```
Thread_101_200 is adding 15050,
  Cumulative Results are [15050]
Thread_1_100 is adding 5050,
  Cumulative Results are [15050, 5050]
Thread_401_500 is adding 45050,
  Cumulative Results are [15050, 5050, 45050]
Thread_301_400 is adding 35050,
  Cumulative Results are [15050, 5050, 45050, 35050]
Thread_201_300 is adding 25050,
  Cumulative Results are [15050, 5050, 45050, 35050, 25050]
Final Result = 125250
```

Part2 (50 Points)

Modify the above program using the wait/notifyAll features. When a thread tries to contribute its results to the shared data, and if it is not this thread's turn, it has to wait. When it is the thread's turn, its contributing result is added to the shared results and all other threads are notified.

Create a package named *cs520.hw6.part2.* Using this package, create the following classes.

2. Create a class named *SharedResults* as follows. The class keeps track of the shared *results*.

    e. The instance (or member) private variable – *results* (ArrayList of integers).
    f. A default constructor that initializes the above data structure.
    g. A void *addToResults* method which takes two arguments, the calling thread's turn and the contributing result that needs to be added to the shared *results*. Implement the wait and notifyAll functionality in this method. Print to the console the thread's turn, the name of the current thread, the value it added, and the shared results data structure. Handle the synchronization issue with this method. Use the size of the data structure to determine if it is the calling thread's turn.
    h. The *getResult* method with no arguments which returns the sum of the values in the shared *results* data structure. Handle the synchronization issue with this method.

4. Create a class named *LongTask* which extends the *Thread* class.

    a. The instance (or member) private variables – *sharedData* (of type *SharedResults*), *start* (integer) , *end* (integer) and *turn* (integer).
    b. A single constructor which takes the above four arguments and stores them in the instance values. Also, create a name for this thread as *Thread_<start>_<end>*

c. In the *run* method, add the integer numbers from *start* to *end* (both inclusive) using a *for* loop. Also, sleep for a random time (up to 10 milliseconds) in each iteration of the loop. After the loop, invoke the *addToResults* method of the shared object and provide this thread's turn and its this accumulated sum.

5. Create a *Test* class to test the following functionality in its *main* method.
   a. Create the *SharedResults* object and assign it to a variable.
   b. Create five *LongTask* objects by passing the above shared object and the *start* ,*end* and *turn* values for each as (1, 100, 0), (101, 200, 1), (201, 300, 2), (301, 400, 3), and (401, 500, 4) respectively.
   c. Start each thread as it is created.
   d. Wait for all the threads to complete using the *join* method.
   e. Print the *result* from the shared object.

Sample Output:

Two sample runs of the program are shown below.

```
Calling Thread's Turn 3, WhoseTurn 0 ... Wait
Calling Thread's Turn 1, WhoseTurn 0 ... Wait
Calling Thread's Turn 0, Thread_1_100 is adding 5050,
  Cumulative Result is [5050]
Calling Thread's Turn 1, Thread_101_200 is adding 15050,
  Cumulative Result is [5050, 15050]
Calling Thread's Turn 3, WhoseTurn 2 ... Wait
Calling Thread's Turn 2, Thread_201_300 is adding 25050,
  Cumulative Result is [5050, 15050, 25050]
Calling Thread's Turn 3, Thread_301_400 is adding 35050,
  Cumulative Result is [5050, 15050, 25050, 35050]
Calling Thread's Turn 4, Thread_401_500 is adding 45050,
  Cumulative Result is [5050, 15050, 25050, 35050, 45050]
Result = 125250
```

```
Calling Thread's Turn 3, WhoseTurn 0 ... Wait
Calling Thread's Turn 4, WhoseTurn 0 ... Wait
Calling Thread's Turn 2, WhoseTurn 0 ... Wait
Calling Thread's Turn 1, WhoseTurn 0 ... Wait
Calling Thread's Turn 0, Thread_1_100 is adding 5050,
  Cumulative Result is [5050]
Calling Thread's Turn 1, Thread_101_200 is adding 15050,
  Cumulative Result is [5050, 15050]
Calling Thread's Turn 2, Thread_201_300 is adding 25050,
  Cumulative Result is [5050, 15050, 25050]
Calling Thread's Turn 4, WhoseTurn 3 ... Wait
Calling Thread's Turn 3, Thread_301_400 is adding 35050,
  Cumulative Result is [5050, 15050, 25050, 35050]
Calling Thread's Turn 4, Thread_401_500 is adding 45050,
  Cumulative Result is [5050, 15050, 25050, 35050, 45050]
Result = 125250
```

Submission:

Create an archive of your Eclipse project using the following steps. Select the HW6_*lastName* project in the Eclipse IDE's *Package Explorer* or the *Navigator* window.

Click *File->Export*. Select the *General->Archive File* option. Click *Next*.

Specify the "*To archive file:"* entry as say, C:\Temp\HW6_*lastName*.zip.

The zip file will be created and stored in the C:\Temp folder.

Submit this zip file as an attachment in the Assignment Section.