Update readme.md
Jonas Solhaug Kaad authored 2 months ago

readme.md 4.53 KiB

Binary File I/O and Threads

You are provided a project comprising 2 packages: `binarystreams` and `threads`

Task a

In the `binarystreams` package, two classes are provided:

- `Species.java` (reduced version of Listing 10.9 in Savitch)
- `WriteSpeciesFileAppend.java` (a version of Listing 10.10, where a new execution of the program tries to append `Species` to the file).

Run the `WriteSpeciesFileAppend.java` two times and use the **same file name** both times. The execution of `WriteSpeciesFileAppend.java` will throw an exception when you try to append `Species` objects in the file.

```
Error opening input file : invalid type code: AC
```

Analyze the code and change it so you don't get the exception.

Hint: *First, check if the file exists. If the file exists, use `writeStreamHeader()` as shown during the lecutre, to reset the header*

Remember to either delete your previous file or use a new one, when checking your solution. Since the previous one will still give you an error

Task b

In the package `threads`, create a task class called `RunnableTask.java` implementing `Runnable`.

- Implement the `Runnable` interface.
- Declare two variables and apply proper encapsulation:
  - `sum(int)`
  - `threadName(String)`
- Create a one argument constructor: `public RunnableTask(String threadName)`
  - Initialize a variable `sum = 0`.
  - Initialize a variable `threadName` using the `String` passed as an argument.
- Implement `run()` method as follows:
  - Create a for loop that executes 10 times.
  - Add the value of the current iteration, to `sum` in each iteration.
  - Print the `threadName`, and the value *(current value of sum)* such that it looks as follows for the first three iterations of the loop:
    - `Thread: A - Current Value: 0`
    - `Thread: A - Current Value: 1`
    - `Thread: A - Current Value: 3`
  - Finally, after the loop is finished, print the sum from each thread such that it looks as follows:
    - `Thread: A - Sum: 45`

Add the following to the `main()` method in `RunnableDemo.java`

- Create three task objects. Pass the name of the thread as an argument (Use names A, B, C for each task)
- Create three threads to perform the 3 tasks
- Start/execute the threads
- Examine output: The **last line** for each of the threads should be:
  - `Thread: A - Sum: 45`
  - `Thread: B - Sum: 45`
  - `Thread: C - Sum: 45`

Remember, the order of Thread A, B and C could be different, but the sum for these threads should be **45**.

Task c

In the package `threads`, create a class `Counter.java`.

- Declare a variable `counter` of type `int`. (remember to apply proper encapsulation)
- Create a Constructor to initialize the variable counter, to be `counter=0;`
- Create a `getCounter()` method for retrieving the value of counter. The method should return the value of the counter.
- Implement an `incrementCounter()` method with the signature `public void incrementCounter()`. The method should increment the value of counter by **2**.
- Implement an `decrementCounter()` method with the signature `public void decrementCounter()`. The method should decrement the value of counter by **1**.

Create a new class, `Task1.java` that extends `Thread`.

- Declare a variable `cr` of type Counter
- Create a 1 argument constructor to initialize the variable `cr`.
- Implement and override the `run()` method, such that it invokes the `incrementCounter()` method 10 times.

Create a new class, `Task2.java` that extends `Thread`.

- Declare a variable `cr` of type Counter
- Create a 1 argument constructor to initialize the variables `cr`.
- Implement and override the `run()` method, such that it invokes the `decrementCounter()` method 10 times.

Add the following to the `main()` method in `ThreadDemo.java`

- Create an object of class `Counter`
- Print the value of counter by using the `getCounter()` method of the `Counter` class.
    - ```
      System.out.println("The value of counter before running threads is: " + counter.getCounter());
      ```
- Create an object of class `Task1` (thread class). Pass the `Counter` object as an argument
- Create an object of class `Task2` (thread class). Pass the `Counter` object as an argument
- Start/execute both threads.
- Write the following statement: `Thread.sleep(1000);` (Also handle exception with a `try-catch`)
- Print the value of counter again by using the `getCounter()` method of the Counter class.
    - ```
      System.out.println("The value of counter after running threads is: " + counter.getCounter());
      ```
- The output should look as follows:

```
The value of counter before running threads is: 0
The value of counter after running threads is: 10
```

```
 1 C:\Users\vivek\.jdks\openjdk-19.0.2\bin\java.exe "-
   javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
   2022.2.1\lib\idea_rt.jar=54164:C:\Program Files\
   JetBrains\IntelliJ IDEA 2022.2.1\bin" -Dfile.encoding
   =UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.
   encoding=UTF-8 -classpath C:\Users\vivek\Downloads\
   binary-file-io-and-threads\target\classes
   binarystreams.WriteSpeciesFileAppendError
 2 Enter output file name.
 3 Vivek
 4 Records sent to file Vivek.
 5 Now let's reopen the file and echo the records.
 6 Name = Calif. Condor
 7 Population = 27
 8 Growth rate = 0.02%
 9
10 Name = Black Rhino
11 Population = 100
12 Growth rate = 1.0%
13
14 Error opening input file Vivek: invalid type code: AC
15
16 Process finished with exit code 0
17
```

```java
package threads;

public class RunnableTask implements Runnable{
    private int sum;
    private String threadName;

    public RunnableTask(String threadName){
        this.sum = 0;
        this.threadName = threadName;
    }

    //Denne metode er drivkraften bag alle Threads, så husk den til Eksamen.
    //Her har jeg lavet en for-lykke til at kunne finde summen af alle værdierne og optælling af sum.
    @Override
    public void run() {
        for(int i = 0; i<10; i++){
            sum+= i;
            System.out.println("Thread:" + threadName + "- Current value " + sum);
        }
        System.out.println("Thread:" + threadName + "- Sum: " + sum);
    }
}

```

```java
package threads;

public class RunnableDemo {

    public static void main(String[] args) {
        //En fejl som jeg lavede i starten var, at jeg direkte gik til threads.
        //Husk altid, at oprette objekter/instanser af den klasse som du ønsker at køre Threads på.
        //Derefter opretter du Thread, ligesom vist fra linje 13 til 15.
        //Derefter bruger du start-metoden eller run-metoden.
        RunnableTask taska = new RunnableTask("A");
        RunnableTask taskb = new RunnableTask("B");
        RunnableTask taskc = new RunnableTask("C");

        Thread threada = new Thread(taska);
        Thread threadb = new Thread(taskb);
        Thread threadc = new Thread(taskc);

        threada.start();
        threadb.start();
        threadc.start();

    }
}

```

```java
package threads;

public class Counter {
    //Det her minder meget om OOP. :)
    //Men det er relevant, at husk at når vi skal bruge værdier, så initialiserer vi variabler og derefter bruger dem i constructoren.
    //Så har vi andre metoder som vi kan bruge.
    private int counter;

    public Counter() {
        this.counter = 0;
    }

    public int getCounter() {
        return counter;
    }

    public void incrementCounter() {
        counter += 2;
    }
    public void decrementCounter() {
        counter--;
    }
}

```

```java
package threads;

public class Task1 implements Runnable{
    Counter cr;

    public Task1(Counter cr){
        this.cr = cr;
    }

    @Override
    public void run(){
        for(int i = 0; i<10; i++){
            //Husk, at bruge metoderne fra de andre klasser på den rigtige måde.
            cr.incrementCounter();
        }
    }
}


```

```java
package threads;

public class Task2 implements Runnable{
    Counter cr;

    public Task2(Counter cr){
        this.cr = cr;
    }
    @Override
    public void run() {
        for(int i = 0; i<10; i++){
            //Husk, at bruge metoderne fra de andre klasser på den rigtige måde.
            cr.decrementCounter();
        }
    }
}


```

```java
package threads;

public class ThreadDemo {

    public static void main(String[] args) throws InterruptedException {
        //Her skal vi bare gøre det samme som vi gjorde i RunnableDemo.
        //Men husk, at følge de instruktioner som er angivet.
        Counter counter = new Counter();
        System.out.println("BEFORE: The value of counter before running threads is: " + counter.getCounter());

        Task1 task1 = new Task1(counter);
        Task2 task2 = new Task2(counter);

        Thread thread1 = new Thread(task1);
        Thread thread2 = new Thread(task2);

        thread1.start();
        thread2.start();

        Thread.sleep(1000);
        System.out.println("AFTER: The value of counter before running threads is: " + counter.getCounter());
    }
}

```