Day 12

Created	@Mar 23, 2021
Created by	Ⓥ VIGNESH J
Tags	Java Thread

FORK & JOIN

- Make use of parallelism
- Attempt to use all available CPU's core
- Accomplished through divide & conquer
- Fork - Divide
- Break the task into smaller independent subtask
- Join - all subtasks are recursively joined into a single result
- The program simply waits until every subtask is executed
- Uses a pool of threads called the ForkJoinPool, which manages worker threads of type ForkJoinWorkerThread.
-

```java
package ga.veee.day12;

import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.RecursiveTask;

public class ForkAndJoinDemo {


    public static void main(String[] args) {
        String arr[] = {"padala", "meghana", "kalpana", "kanpur", "megha", "raghul", "raghul", "mohan", "raghul", "meghana"};
        int count = 0;
        for (String s : arr) {
            if (s.equalsIgnoreCase("raghul")) {
                count++;
            }
        }
        System.out.println("Count of raghul...:" + count);

        ForkJoinPool fjp = ForkJoinPool.commonPool();

        MyTask task1 = new MyTask(arr, 0, 3, "raghul");
        MyTask task2 = new MyTask(arr, 3, 5, "raghul");
        MyTask task3 = new MyTask(arr, 5, 7, "raghul");
        MyTask task4 = new MyTask(arr, 7, 9, "raghul");

        int result1 = fjp.invoke(task1);
        int result2 = fjp.invoke(task2);
        int result3 = fjp.invoke(task3);
        int result4 = fjp.invoke(task4);

        int total = result1 + result2 + result3 + result4;
        System.out.println("Total raghuls are...:" + total);

    }
}

class MyTask extends RecursiveTask<Integer> {
    String arr[];
    int start, end;
    String searchString;

    public MyTask(String arr[], int start, int end, String searchString) {
        this.arr = arr;
        this.start = start;
        this.end = end;
        this.searchString = searchString;
    }

    @Override
    protected Integer compute() {
        int count = 0;
        for (int i = start; i < end; i++) {
            if (arr[i].equalsIgnoreCase(searchString)) {
                count++;
            }
        }
        return count;
    }
}
```

https://fluvid.com/videos/detail/w6e8gco-jMcnG994e#.YFl6RYoicws.link

Daemon Threads

- Daemon threads are low priority threads which run in background
- User threads are high priority threads which run in foreground
- User Thread or Non-Daemon are designed to do specific or complex task
- Daemon threads are used to perform supporting tasks.
- A daemon thread is a low priority thread that is considered doing some tasks in the background like handling requests or various cronjobs that can exist in an application.

```
package ga.veee.day12;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class DeamonThreadDemo {
    public DeamonThreadDemo() {
//    new Thread(new Runnable() {
//      @Override
//      public void run() {
//        while(true) {
//          System.out.println("child thread.........");
//        }
//      }
//    }).start();
        ExecutorService es = Executors.newFixedThreadPool(1);
        es.execute(() -> {
            while (true) {
                System.out.println("child thread....");
            }
        });
    }

    public static void main(String[] args) {
        new DeamonThreadDemo();
        System.out.println("main thread started....");
        Thread.currentThread().setDaemon(true);
        try {
            Thread.sleep(2000);
        } catch (Exception e) {
        }
        System.exit(1);
        System.out.println("hello");
    }
}
```

Garbage Collection

```
package ga.veee.day12;

import java.lang.ref.SoftReference;
import java.lang.ref.WeakReference;
public class GarbageDemo {
    public static void main(String[] args) {
        Runtime r=Runtime.getRuntime();
        System.out.println("Before Tathas birth...:"+r.freeMemory());
        GrandFather tatha=new GrandFather();
        System.out.println("After Tathas birth....:"+r.freeMemory());

        //WeakReference wf=new WeakReference(tatha);// this is also a way to certify the object for garbage collection
        SoftReference soft=new SoftReference(tatha);
        tatha=null; //the object will still live in memory

        //  System.out.println(tatha.gold);
        System.out.println("After Tathas death....:"+r.freeMemory());
        System.out.println("tathas body is still lying in house..............");
        //tatha=null;
        r.gc();//this will request the object to be removed..
        //tatha=(GrandFather)soft.get();
        //System.out.println(tatha.gold);
        System.out.println("After kariyam.........."+r.freeMemory());

    }
}
class GrandFather{
    String space;
    String gold="under the tree......";
    public GrandFather() {
        for(int i=0;i<10000;i++) {
            space=new String(".........."+i);
        }
    }
    @Override
    protected void finalize() throws Throwable {
        System.out.println("finalize method called...");
    }
}
```

Best Free Online Video Recording & Screen Capture Software | Fluvid

Fluvid is the best all-in-one online screen capture & video recording software that is available for free. Fluvid helps you record, edit, communicate & share your video messages in the most simplest way.

https://fluvid.com/videos/detail/gk_qEcDnp_FaXOrQM#.YFmZD-MiUUY.link

Generic

-

```
package ga.veee.day12;

public class GenericsDemo {
    public static void main(String[] args) {
        PaintBrush<Paint> brush=PaintContainer.getPaintBrush();
        Paint paint=(Paint)brush.obj;
        System.out.println(brush.getObj());
        PaintBrush<Water> waterBrush=WaterContainer.getPaintBrush();
        Water water=(Water)waterBrush.getObj();
        System.out.println(waterBrush.getObj());
    }
}
class PaintContainer {
    public static PaintBrush<Paint> getPaintBrush() {
        PaintBrush<Paint> pb=new PaintBrush<>();
        Paint obj=new RedPaint();
        pb.setObj(obj);
        return pb;
    }
}
class WaterContainer{
    public static PaintBrush<Water> getPaintBrush() {
        PaintBrush<Water> pb=new PaintBrush<>();
        Water obj=new Water();
        pb.setObj(obj);
        return pb;
    }
}
abstract class Paint{

}
class RedPaint extends Paint{

}
class BluePaint extends Paint{

}
class PaintBrush<T>{
    //Paint paint;
    //Object obj;//generic
    //real generics is
    T obj;
    public T getObj() {
        return this.obj;
    }
    public void setObj(T obj) {
        this.obj=obj;
    }
}
class Water {}
```

```
package ga.veee.day12;

public class GenericMethodTest {
    // generic method printArray
    public static <E> void printArray(E[] inputArray) {
        // Display array elements
        for (E element : inputArray) {
            System.out.printf("%s ", element);
        }
        System.out.println();
    }

    public static void main(String args[]) {
        // Create arrays of Integer, Double and Character
        Integer[] intArray = {1, 2, 3, 4, 5};
        Double[] doubleArray = {1.1, 2.2, 3.3, 4.4};
        Character[] charArray = {'H', 'E', 'L', 'L', 'O'};
        System.out.println("Array integerArray contains:");
        printArray(intArray);   // pass an Integer array
        System.out.println("\nArray doubleArray contains:");
        printArray(doubleArray);   // pass a Double array
        System.out.println("\nArray characterArray contains:");
        printArray(charArray);   // pass a Character array
    }
}
```

Annotation

- Retention Policy
    - Compile time
    - Run time

```
package ga.veee.day12.annotation;

import java.lang.reflect.Field;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;

@Retention(RetentionPolicy.RUNTIME)
@interface In {
    String value() default "helloworld";
}

public class GenericsDemo {
    public static void main(String[] args) throws Exception {
        PaintBrush<Paint> brush = PaintContainer.getPaintBrush();
        Paint paint = (Paint) brush.obj;
        System.out.println(brush.getObj());
        PaintBrush<Water> waterBrush = WaterContainer.getPaintBrush();
        Water water = (Water) waterBrush.getObj();
        System.out.println(waterBrush.getObj());
    }
}

class PaintContainer {
    public static PaintBrush<Paint> getPaintBrush() throws Exception {
        PaintBrush<Paint> pb = new PaintBrush<>();
        Paint obj = new RedPaint();
        Field field = pb.getClass().getDeclaredField("obj");
        field.setAccessible(true);
        In in = field.getAnnotation(In.class);
        if (in != null) {
            pb.setObj(obj);//dependency injection
        }
        return pb;
    }
}

class WaterContainer {
    public static PaintBrush<Water> getPaintBrush() throws Exception {
        PaintBrush<Water> pb = new PaintBrush<>();
        Water obj = new Water();
        Field field = pb.getClass().getDeclaredField("obj");
        field.setAccessible(true);
        In in = field.getAnnotation(In.class);
        if (in != null) {
            String s = in.value();
            System.out.println("The value..is:" + s);
            pb.setObj(obj);//dependency injection}
        }
        return pb;
    }
}

abstract class Paint {

}

class RedPaint extends Paint {

}

class BluePaint extends Paint {

}

class PaintBrush<T> {
    //Paint paint;
    //Object obj;//generic
    //real generics is
    @In//(value="helloworld")
            T obj;

    public T getObj() {
        return this.obj;
    }

    public void setObj(T obj) {
        this.obj = obj;
    }
}

class Water {
}
```

String

- Buffer
    - Used to create mutable string
    - The StringBuffer class in java is same as String class except it is mutable
- Builder
    - Used to create mutable string
    - StringBuilder class is same as StringBuffer class except that it is non synchronised.
- Joiner
    - From java.util package
    - It is used to construct a sequence of characters separated by a delimiter.
    -

```
package ga.veee.day12;

import java.util.StringJoiner;
```

```
public class StringDemo {
    public static void main(String[] args) {
        String s = "hello world";
        String ss = new String("hello world");

        StringJoiner joinstr = new StringJoiner(",");
        joinstr.setEmptyValue("its a empty string..");
        System.out.println(joinstr);
        joinstr.add("hello");
        joinstr.add("hai");
        System.out.println(joinstr);

        joinstr = new StringJoiner(",", "[", "]");
        joinstr.add("hello");
        joinstr.add("hai");
        System.out.println(joinstr);

        StringBuffer sbf = new StringBuffer();//synchronized
        sbf.append("hello");
        StringBuilder sb = new StringBuilder();//non synchronized..
        sb.append("hai");
    }
}
```

Vargs, Formatting

```
package ga.veee.day12;

/*
 * %- denoting the start of the formatting instructions
 * [flag]- - pad on right + pad on left
 * [width] - size of padding
 * [.precision]-this if for floating numbers.
 * d/s/f - denoting integer,string and float
 *
 */
public class StringFormatDemo {
    public static void main(String[] args) {
        System.out.println("hello\nworld");

        System.out.printf("%d students given by..%s and there average marks is...%.3f", 20,"coda",82.456666);

        System.out.printf("\n%10s%-10s%s","Column1","Column2","Column3");

    }

    public static void met(int... i) {//varargs
        for(int num:i) {
            System.out.println(num);
        }
    }
}
```

Best Free Online Video Recording & Screen Capture Software | Fluvid

Fluvid is the best all-in-one online screen capture & video recording software that is available for free. Fluvid helps you record, edit, communicate & share your video messages in the most simplest way.

https://fluvid.com/videos/detail/xLOkKcR9eGh17z5o9#.YFnMqnqjcTE.link

enum

- Enum is a way to declare constant
- Can be used as a Object

-

```
package ga.veee.day12;

public class EnumDemo {
    public static void main(String[] args) {
        Cars c;
        c = Cars.honda;
        met(c);
        Cars cc[] = c.values();
        for (Cars car : cc) {
            System.out.println(car + " price is : " + car.getPrize());
        }

    }

    public static void met(Cars c) {
        switch (c) {
            case maruthi: {
                System.out.println("The car is maruti.........");
                break;
            }
            case suzuki: {
                System.out.println("its suzuki..........");
                break;
            }
            default: {
                System.out.println("default..........ambi");
            }
        }
    }
}

enum Cars {
    maruthi(1000), suzuki, honda, nissan;
    int prize;

    Cars() {
        System.out.println("cons called....");
    }

    Cars(int prize) {
        System.out.println(this.name() + " car price is : " + prize);
        this.prize = prize;
    }

    public int getPrize() {
        return this.prize;
    }
}
```

Time

-

```
package ga.veee.day12;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.time.Month;
import java.time.temporal.ChronoUnit;

public class DateDemo {
    public static void main(String[] args) {
        LocalDateTime currentTime = LocalDateTime.now();
        System.out.println(currentTime);

        LocalDate date1 = currentTime.toLocalDate();
        System.out.println("date1: " + date1);

        Month month = currentTime.getMonth();
        int day = currentTime.getDayOfMonth();
        int seconds = currentTime.getSecond();

        System.out.println("Month: " + month + "day: " + day + "seconds: " + seconds);

        LocalDateTime date2 = currentTime.withDayOfMonth(10).withYear(2012);
        System.out.println("date2: " + date2);

        //12 december 2014
        LocalDate date3 = LocalDate.of(2014, Month.DECEMBER, 12);
        System.out.println("date3: " + date3);

        //22 hour 15 minutes
        LocalTime time = LocalTime.of(22, 15);
        System.out.println("Time:" + time);

        //parse a string
        LocalTime time2 = LocalTime.parse("20:15:30");
        System.out.println("Time2: " + time2);

        LocalDate today = LocalDate.now();
        System.out.println("Current date: " + today);

        //add 1 week to the current date
        LocalDate nextWeek = today.plus(1, ChronoUnit.WEEKS);
        System.out.println("Next week: " + nextWeek);

        //add 1 month to the current date
        LocalDate nextMonth = today.plus(1, ChronoUnit.MONTHS);
        System.out.println("Next month: " + nextMonth);

        //add 1 year to the current date
        LocalDate nextYear = today.plus(1, ChronoUnit.YEARS);
        System.out.println("Next year: " + nextYear);

        //add 10 years to the current date
        LocalDate nextDecade = today.plus(1, ChronoUnit.DECADES);
        System.out.println("Date after ten year: " + nextDecade);
    }
}
```

Assignment

- Start Date & Time
- Travel Speed
- Distance
- No of hours working per day
- Holiday list [Sunday, JAN1,JAN26,AUG15,OCT2]
- Calculate the date and time of delivery