95

Written Test

Sep 17, 2018

Name: Zihao Liu

Part 1: Coding

Define a 3 v 3 soccer game program, by implementing below 3 classes Player, Timer and Match, following the requirements below for each class:

- Ball (No need to implement this class)
  (1) A ball object will be shared across all the players in one match (Hint: it can be used as a lock object).

- **Player** :
  (1) The object should be **able to be cloned** for new player's creation.
  (2) And each Player object can be used to start its own **thread**.
  (3) The ball can be hold by only one player at a time, the ball holding player will hold the ball for 1 sec, then print out a message with his ID in it, eg. like "Number 2 owns the ball...", then check if time is out (By using the Timer object defined in class Match), if not, pass the ball out, and one of the other player will get the ball; otherwise, quit the match by finish the thread's execution.

  Fields:
  - id : int
  - name : String

- **Timer**
  (1) A timer object can be used to start a **thread**.
  (2) It contains a flag to indicate if the match is completed or not.
  (3) Each match will play 1 min, so once the timer thread is running for a min, the flag get toggled.

  Fields:
  - private boolean complete;

  Functions:
  - public boolean isComplete() : this function indicate if time is up for the match, by checking the "complete" field.

- **Match**
  (1) There should have only one match object defined in the JVM, so this should be a **Java singleton** class.
  (2)
  Fields:
  - Object ball;
  - int id; // the match ID
  - List<Player> teamA; // number 1-5
  - List<Player> teamB; // number 6-10
  - Timer timer;

  Functions:
  - Create and clone all players when match object is created.
  - void startMatch(). This functions will start the timer, and also, all players from both team will start to play.

Part 2: DataBase

For the above match application, please use SQLs to
(1). Create below 2 tables structure.
(2). Please use SQL to insert 1 row for each table only. (Data below are only sample dummy data, you can use any other testing data.)
(3). **(No point loss if you cannot complete this, but extra 3 points will be added if you can do it right)** Foreign key creation, ID field in table Player is been used as a foreign key in table MatchHistory, column PlayerId.

Table **Player**

ID (Player ID, Integer, not null)	**NAME(Player name, String, max length 30)**	**TEAM (Team name, String, max length 10)**
1	Jane	R
2	Wing	R
3	Sherry	R
4	Karen	M
5	Monette	M
6	Ashley	M

Table **MatchHistory**

ID (Match ID, Integer, not null)	**PlayerId (Player ID, Integer, not null)**	**INFO (String, max length 50)**
1	1	Number 1 owns the ball...
1	3	Number 3 owns the ball...
1	4	Number 4 owns the ball...
1	1	Number 1 owns the ball...
1	2	Number 2 owns the ball...

(2). Use 1 query (may contains sub query) to find out the MVP players (The player who owns the ball at most in one match, maybe more than 1 players will be found), and display all her information. Below columns are required:

(Result table with sample dummy data:)

PlayerID	**Name**	**Team**	**MatchID**
1	Jane	R	1
3	Sherry	R	2
6	Ashley	M	2
3	Sherry	R	3

Part 3: Code Output

1. Are lines from below code legal or not legal? Use T/F (T for legal, F for not legal) to mark on each lines below.

```
public class A {
        class B {                          T
                static void foo() { }      T
                C c=new C();               T
        }
        static class C {                   T
                static void foo() { }      T
                void foo2() { }            F
                B b=new B();               F
        }
        static void foo1() {               T
                new B();                   F
                new C();                   T
        }
        void foo2() {                      T
                new B();                   T
                new C();                   T
        }
}
```

2.

```
class Base {
    int x = 3;
    public void foo() {
        System.out.println(x + " in Base");
    }
    Base() {
        foo();
    }
}
class Sub extends Base {
    int x = 5;
    public void foo() {
        System.out.println(x + " in Sub");
    }
    Sub() {
        foo();
    }
}
Base b = new Sub();
```

What will be the output for above code?

in Sub
0 ~~In Base~~
5 in Sub.

Part 4: Questions & Answers

1. In Java, what's the differences among interface, abstract class and concrete class?

2. Please explain the Java String pool, and what intern() function does in Java.

3. Please explain the internal structure of Java HashMap, how put() function works, and when will this function may cause a dead lock?

: Zihao Liu.

```
public class Player implements Cloneable, Runnable{    // Player
      private int id;
      private String name;
      private Object ball;
      public Player(){}
      public Player(int id, String name, Object ball){
            this.id= id
            this.name =name;
            this.ball =ball;
      }
      @Override
       public Player clone() throws Exception{
              return (Player) super.clone();
       }
      @Override.
       public void run(){
          synchronized(ball){
          try {
               Thread.sleep(1000);
            System.out.println("Number " +id +" owns the ball..");
               ball.wait();
          } catch (Exception e){
               e.printStackTrace();
          }}
       }
}
public class Timer implements Runnable {    // Timer
         private boolean complete;
         public Timer(){ complete=false;}
         public boolean isComplete(){
                // some code here

              return complete;
         }
```

Player

while loop -1

need to notify other before wait

-1

```
@Override
public void run() {
        try {
          Thread.Sleep(60000);
          complete = true.
        } catch (Exception e){
          e.printStackTrace();
        }
  }
}

public class Match {  // Have no time to write getter and setter, but It should have
  private Object ball;
  private int id;
  private List<Player> teamA;
  private List <Player > ~~tam~~ teamB;
  private Timer timer;
  private static Match match = null;
  private Match () {}
  private Match ( Object ball, int id, List<Player> teamA List<Player> teamB, Timer timer){
        this.ball = ball;
        this.id = id;
        this.teamA = teamA;
        this.teamB = teamB;
        this.timer = timer;
  }
  public static ~~void~~ ~~Match~~ Match getInstance() {
        if (match == null) {
              synchronized (match){
                    if (match == null){
                          match = new Match();
              }}}}   return match;
```

```
.ic  void   startMatch() throws Exception {

    match.setBall(new Object());
    List<Player> ~~List~~ teamA= new ArrayList();
    for (int i=1;i<=5;i++){
                                        Player(i, i+"teamA", match.getBall()).clone());
      ~~match.set~~ ~~teaA~~ teamA.add(new ~~Player(i,i+"teamA").clone());~~
    }
    List <Player>  teamB= new ArrayList();
    for (int i=6 ; i <= 10; i++){
                                        ,match.getBall()
          teaB.add(new Player(i,"teamB").clone());
    }
    match.setTeamA(teamA);
    match.setTeamB(teamB);
    match.setTimer(new Timer());
    ~~while (match.getTimer().sto~~
     new Thread(match.getTimer()).start();
     while (!match.getTimer().isComplete()){
          ~~match.getBall().notifyAll();~~
          for (Player p: teamA){
               match.getBall().notify();
               new Thread(p).start();
          }
          for (Player P: teamB){
               match.getBall().notify();
               new Thread(P).start();
          }
     }
}
```

why still clone?

-1

Can you notify in here?

Part 2:

(1)
```
create table Player (
    ID int, not null,
    NAME varchar2(30),
    TEAM varchar2(10)
)
```

```
create table MatchHistory (
    ID int not null,
    PlayerID int not null,
    INFO varchar2(50)
}
```

(2)
```
insert into Player
values (1, 'Jane', 'R');
```

```
insert into MatchHistory
values (1, 1, 'Number 1 owns the ball.');
```

alter. but should use create

(3)
```
alert table MatchHistory
~~modify~~ add ( constraint MH_PID_FK foreign key (PlayerId)
        references Player(ID) ).
```

Part 3. 1. T T   T F F   F T   T T.

2. ~~0 inBase 5 inSub~~ 0 in Sub 5 in Sub.

Part 4. Interface: 1. All functions are abstract without body.
2. Only allow public final static fields, and It's default.
3. can't create object.
4. can extends interface.

Abstract class: 1. can have abstract and concrete functions.
2. allow all fields.
3. can extends concrete class and implements itertaces.
4. If It has beedn overrided, It's abstract functions must be overrided too.

Concrete class: 1. ~~can ha~~ only have concrete functions
2. allow all fields.
3. can creat objects.
4. can extends all class, ~~and im~~ (but only one), implements all interfaces.

string Pool is used to store string in PG. In SP. the same String can only ~~exsit~~ one. SP use FlyWeight Design Pattern, So the string is shared by all references who own same string.

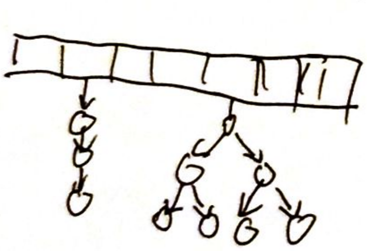
The intern() function can copy a string and insert it into String Pool. ~~like~~ like the code below.

check, before copy

```
String a= "abc";
String b= new String("abc");
String c = b.intern()
System.Out.println( a==b); // false
System.out.println(b==c); // false
System.out.println (a==c); // true.
```

3. the internal structure of HashMap is include, nodes, nodes list, nodes tree

an array



When Run put function, Hash Map will run in below steps.

1. creat node<K,V>
2. calculato key's hash code(). and run hash() to get hash number
3. use hash number to locate the bins.
4. If bin is Empty
    4.1 check size of HashMap.
        4.11 If it's over threshold, resizeit.
        4.12 ~~put node in this bin~~ put it in this bin
5. If bin is not Empty
    5.1 use equals () to find whether there is same key.
        5.11 If yes, ~~run 4.1. check the list size.~~ replace it with new values

~~it's the size is over treeifY_threshold, change it into tree.~~

~~other: After resizing, replace value with old val new values.~~

~~6. If~~ 5.12 if there is no same key node, run 4.1, if the ~~so~~ size is over treeifY_THRESHOLD, change it into a tree. ~~kick the bin out~~ then kick the node out and put node there.

Dead lock.

when two thread try to put two nodes ~~at~~ in HushMap at the same time. there may cause ~~a~~ infinite loop. error.

when 2 thread trying to reeize at same time.

—1