NOME: *COGNOME:* *MATRICOLA:*
SUPERATO ESAMI PROPEDEUTICI (SI/NO): *FIRMA:*

Programmazione Concorrente e Distribuita

Esame scritto del 22 Marzo 2012
Versione K

Esercizio 1 (5 punti)

Si consideri il codice seguente.

```
1  public class Redwood extends Tree {
2          public static void main(String[] args) {
3                  new Redwood().go();
4          }
5
6          void go() {
7                  go2(new Tree(), new Redwood());
8                  go2((Redwood) new Tree(), new Redwood());
9          }
10
11         void go2(Tree t1, Redwood r1) {
12                 Redwood r2 = (Redwood)t1;
13                 Tree t2 = (Tree)r1;
14         }
15 }
16
17 class Tree { }
```

Quale è il risultato?

A Viene sollevata un'eccezione a runtime.

B Il codice compila ed esegue senza output.

C La compilazione fallisce con un errore alla linea 7.

D La compilazione fallisce con un errore alla linea 8.

E La compilazione fallisce con un errore alla linea 12.

F La compilazione fallisce con un errore alla linea 13.

Esercizio 2 (12 punti)

Si considerino le seguenti definizioni.

```
1  class D{
2          String x="pippo";
3
4          D(String s){
5                  x=s;
6          }
7
8          public synchronized void set(String str){
9                  x=str;
10         }
11
12         public synchronized String get(){
13                 return x;
14         }
15 }
16
17 class A{
18         private D s;
19
20         A(String st){
21                 s=new D(st);
22         }
23
24         A(D d){
25                 s=d;
26         }
27
28         public synchronized void set(String str){
29                 s.x=str;
30         }
31
32         public synchronized String get(){
33                 return s.x;
34         }
35
36         synchronized void f(int k) throws InterruptedException{
37                 System.out.println(Thread.currentThread().getName()
38                         + " In f() s="+s.x);
39
40                 for(int i=0; i<100000; i++);
41                 if(! get().equals(Thread.currentThread().getName()))
42                         set(Thread.currentThread().getName());
43
44                 for(int i=0; i<100000; i++);
45                 System.out.println(Thread.currentThread().getName()
46                         + " Fin f() s="+s.x);
47         }
48
49         synchronized void g(int k) throws InterruptedException{
50                 System.out.println(Thread.currentThread().getName()
51                         + " In g() s="+s.x);
52
53                 for(int i=0; i<100000; i++);
54                         if(! s.get().equals(Thread.currentThread().getName()))
55                                 s.set(Thread.currentThread().getName());
56
57                 for(int i=0; i<100000; i++);
58                         System.out.println(Thread.currentThread().getName()
```

```
59                                      + " Fin g() s="+s.x);
60          }
61  }
62
63  class T1 extends Thread{
64          private A a;
65
66          T1(String nome, A a){
67                  super(nome);
68                  this.a=a;
69          }
70
71          public void run(){
72                  try{
73                          for(int i=0; i<10; i++) a.f(i);
74                          C.flag++;
75
76                          while(C.flag<2);
77                          for(int i=0; i<10; i++) a.g(i);
78                  }catch(InterruptedException e){}
79                  C.flag++;
80          }
81  }
82
83  public class C{
84          public static int flag=0;
85
86          public static void main(String[] arg){
87                  T1 t1=new T1("Alice", new A(new String("pluto")));
88                  T1 t2=new T1("Bob", new A(new String("pluto")));
89                  t1.start();
90                  t2.start();
91
92                  while(flag<4){
93                          try{
94                                  Thread.sleep(1000);
95                          } catch(Exception e){}
96                  }
97                  D d=new D("topolino");
98                  T1 t3=new T1("Carl", new A(d));
99                  T1 t4=new T1("David", new A(d));
100                 t3.start();
101                 t4.start();
102         }
103 }
```

Per ciascuna delle seguenti affermazioni, indicare se si tratta di un'affermazione VERA o FALSA.

A E' possibile che in una qualche esecuzione del programma si verifichi il seguente interleaving di stampe consecutive.

```
Alice In f() s=Alice
Bob In f() s=Bob
Alice Fin f() s=Alice
Bob Fin f() s=Bob
```

B In ogni esecuzione del programma si verifica il seguente interleaving di stampe consecutive.

```
Alice In f() s=Alice
Bob In f() s=Bob
Alice Fin f() s=Alice
Bob Fin f() s=Bob
```

C E' possibile che in una qualche esecuzione del programma si verifichi il seguente interleaving di stampe consecutive.
```
Alice In f() s=Alice
Bob In g() s=Bob
Alice Fin f() s=Alice
```

D E' possibile che in una qualche esecuzione del programma si verifichi il seguente interleaving di stampe consecutive.
```
Alice In g() s=Alice
Bob In g() s=Bob
Bob Fin g() s=Bob
Alice Fin g() s=Alice
Carl In f() s=topolino
David In f() s=topolino
```

E E' possibile che in una qualche esecuzione del programma si verifichi il seguente interleaving di stampe consecutive.
```
Carl In f() s=topolino
Carl Fin f() s=Carl
David In f() s=topolino
David Fin f() s=David
```

F In ogni esecuzione del programma non compare mai la stampa della stringa
`Alice Fin g() s=Bob`.

G E' possibile che in una qualche esecuzione del programma si verifichi il seguente interleaving di stampe consecutive.
```
David In f() s=David
Carl In f() s=Carl
David Fin f() s=David
Carl Fin f()=Carl
```

H E' possibile che in una qualche esecuzione del programma si verifichi il seguente interleaving di stampe consecutive.
```
David In f() s=David
Carl In f() s=David
David Fin f() s=David
Carl Fin f() s=Carl
```

I In ogni esecuzione del programma non compare mai la stampa della stringa `Carl In g() s=David`.

J Se nessuno dei metodi della classe A fosse sincronizzato, il programma si comporterebbe in maniera diversa.

K I thread `t1` e `t2` terminano sempre prima dei thread `t3` e `t4`.

L Il `main` thread è sempre l'ultimo thread a terminare.

Esercizio 3 (8 punti)

Si consideri il seguente codice che crea e avvia due threads:

```
1  public class C{
2          public static void main(String[] arg){
3                  T t1=new T("t1");
4                  T t2=new T("t2");
5
6                  t1.setAltro(t2);
7                  t2.setAltro(t1);
8                  t1.start();
9                  t2.start();
10         }
11 }
12
13 class T extends Thread{
14         String nome;
15         boolean libero=true;
16         T altro;
17
18         T(String s){
19                 nome=s;
20         }
21
22         void setAltro(T t){
23                 altro=t;
24         }
25
26         public void run(){
27                 .....
28         }
29 }
```

Per ognuna delle seguenti definizioni del metodo `run()` della classe `T`, dire se le affermazioni riportate sono vere o false.

```
1  public void run(){
2          try{
3                  synchronized(this){
4                          while(altro.libero==false){
5                                  try{
6                                          libero=true;
7                                          wait();
8                                          libero=false;
9                                  }catch(InterruptedException e){ }
10                         }
11                         libero=false;
12                         synchronized(altro){
13                                 altro.libero=false;
14                                 System.out.println("sezione critica di "+nome);
15                                 altro.libero=true;
16                                 altro.notify();
17                         }
18                         libero=true;
19                 }
20         }catch(Exception e){}
21 }
```

a Il programma può andare in deadlock.

b Se un thread si sospende con wait, non verrà mai risvegliato.

```
1  public void run(){
2          try{
3                  synchronized(this){
4                          libero=false;
5                          while(altro.libero==false){
6                                  try{
7                                          libero=true;
8                                          wait();
9                                          libero=false;
10                                 }catch(InterruptedException e){ }
11                         }
12                         synchronized(altro){
13                                 altro.libero=false;
14                                 System.out.println("sezione critica di "+nome);
15                                 altro.libero=true;
16                         }
17                         libero=true;
18                         notify();
19                 }
20         }catch(Exception e){}
21 }
```

a Il programma termina sempre producendo in output la stampa `sezione critica di t1 sezione critica di t2` oppure `sezione critica di t2 sezione critica di t1`.

b Il programma non produce mai in output la stampa `sezione critica di t1 sezione critica di t2` oppure `sezione critica di t2 sezione critica di t1`.

Esercizio 4 (5 punti)

Si consideri il codice seguente.

```
1  class Test {
2          public static void main(String [] args) {
3                  printAll(args);
4          }
5          public static void printAll(String[] lines) {
6                  for(int i=0;i<lines.length;i++){
7                          System.out.println(lines[i]);
8                          Thread.currentThread().sleep(1000);
9                  }
10         }
11 }
```

Il metodo statico `Thread.currentThread()` ritorna un reference all'oggetto Thread che è in esecuzione. Qual'è il risultato di questo codice?

A Ciascuna String nell'array `lines` verrà stampata, con una pausa di esattamente un secondo tra una e l'altra.

B Ciascuna String nell'array `lines` verrà stampata, senza pause tra una e l'altra perchè questo metodo non è eseguito in un Thread.

C Ciascuna String nell'array `lines` verrà stampata, e non vi è garanzia che ci sia una pausa perchè `currentThread()` potrebbe non selezionare questo thread.

D Questo codice non compila.

E Ciascuna String nell'array `lines` verrà stampata, con una pausa di almeno un secondo tra una e l'altra.