

Resolução dos exercícios da aula T03a

Exercício 1:

Pretende-se construir um programa que permita adicionar arrays de grande dimensão. Para isso, vamos dividir os arrays a somar em várias porções menores e somar cada porção numa thread independente.

a) Construir o código de uma thread, ThreadSoma, que receba como parâmetros as “referências” de 3 arrays, A, B e C e 2 valores inteiros, p e u. A thread deverá fazer a soma dos arrays, A e B, desde a posição p até à posição u-1, colocando o resultado em C.

$$C[i] = A[i] + B[i] \quad \text{com } i = p, p+1, p+2, \dots, u-1.$$

```
public class ThreadSoma extends Thread {  
    int[] A, B, C;  
    int p, u;  
    public ThreadSoma( int[] A, int[] B, int[]C, int p, int u) {  
        super();  
        this.A = A;  
        this.B = B;  
        this.C = C;  
        this.p = p;  
        this.u= u;  
        start();  
    }  
    public void run(){  
        for (int i = p; i < u; i++) {  
            C[i] = A[i] + B[i];  
        }  
    }  
}
```

b) Construa um programa onde deverá criar dois arrays de inteiros com valores aleatórios. Os arrays, A e B, devem ter a mesma dimensão. Para somar os dois arrays, colocando a soma num array, C, com a mesma dimensão, divida-os em dois e cada metade deverá ser somada por uma instância da classe ThreadSoma que criou na alínea anterior.

```
public class Teste {  
    public static void main(String[] args) {  
        int dim = 1000000;  
        int[] A = new int [dim];  
        int[] B = new int [dim];  
  
        for (int i = 0; i < A.length; i++) {  
            A[i] = (int)(Math.random()*100);  
            B[i] = (int)(Math.random()*100);  
        }  
        int [] C = new int[dim];  
    }  
}
```

b)

```
ThreadSoma ts1 = new ThreadSoma (A, B, C, 0, dim/2);
```

```
ThreadSoma ts2 = new ThreadSoma (A, B, C, dim/2, dim);
```

```
// Experimente mostrar o último valor do array resultado:
```

```
System.out.println (C[dim-1]);
```

O que obtém?

O que aconteceu?

Explore o método **join** da classe Thread e tente acabar o exercício 1.

Sincronização de Threads exemplos: (ver T03a)

```
public class Exemplo2 {
```

```
    private int a = 1, b = 2;
```

```
    public void ab()  
    { a = b; }
```

```
    public void ba()  
    { b = a; }
```

```
    public int getA() { return a; }
```

```
    public void setA(int a) { this.a = a; }
```

```
    public int getB() { return b; }
```

```
    public void setB(int b) { this.b = b; }
```

```
}
```

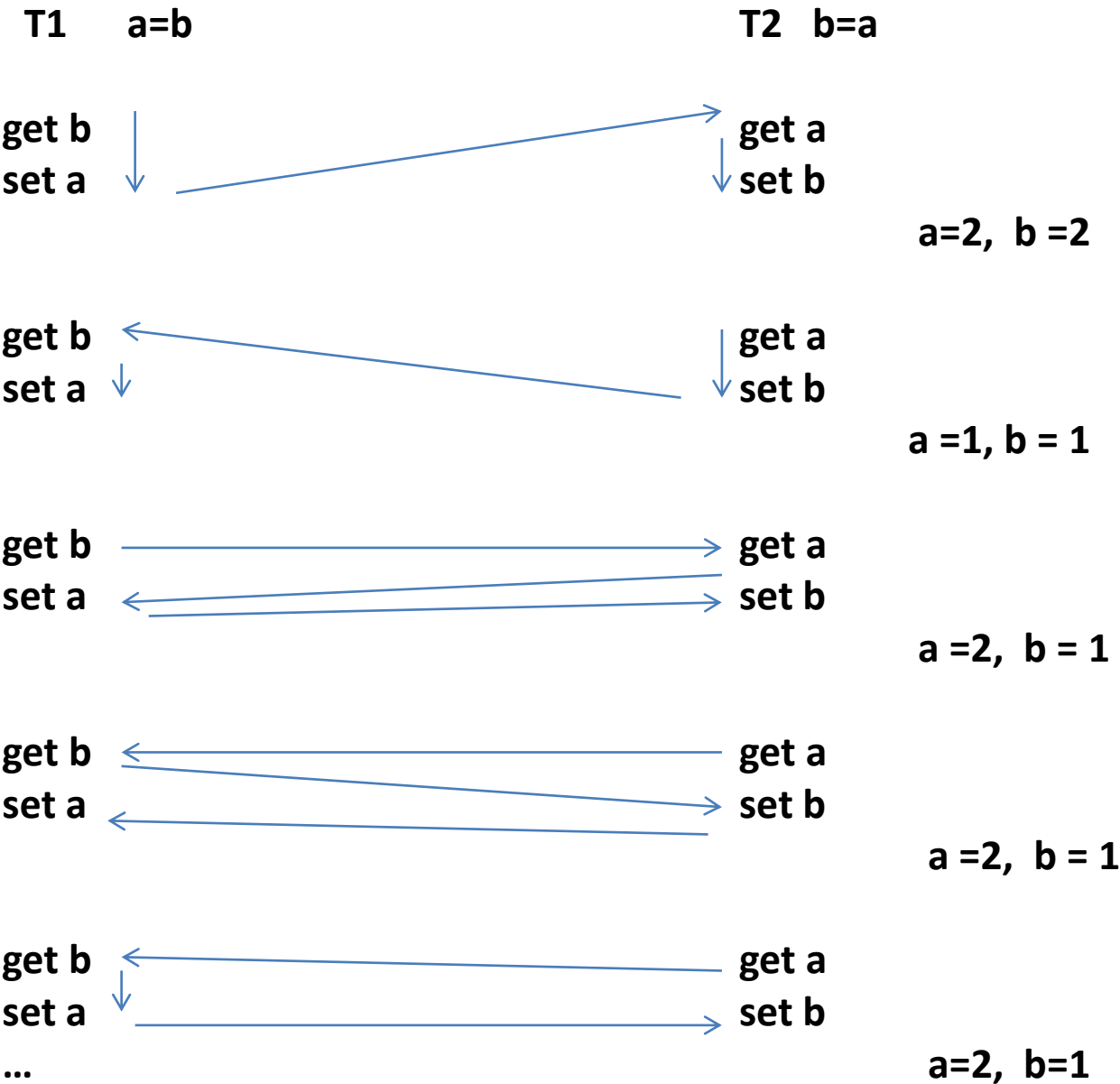
```
public class T1 extends Thread{
    Exemplo2 X;
    public T1(Exemplo2 x){
        super();
        X=x;
        start();
    }
    public void run(){
        X.ab();
    }
}
```

```
public class T2 extends Thread{
    Exemplo2 X;
    public T2(Exemplo2 x){
        super();
        X=x;
        start();
    }
    public void run(){
        X.ba();
    }
}
```

```
public static void main(String[] args) {
    Exemplo2 e = new Exemplo2();
    T1 t1 = new T1(e);
    T2 t2 = new T2(e);
```

O que pode acontecer?

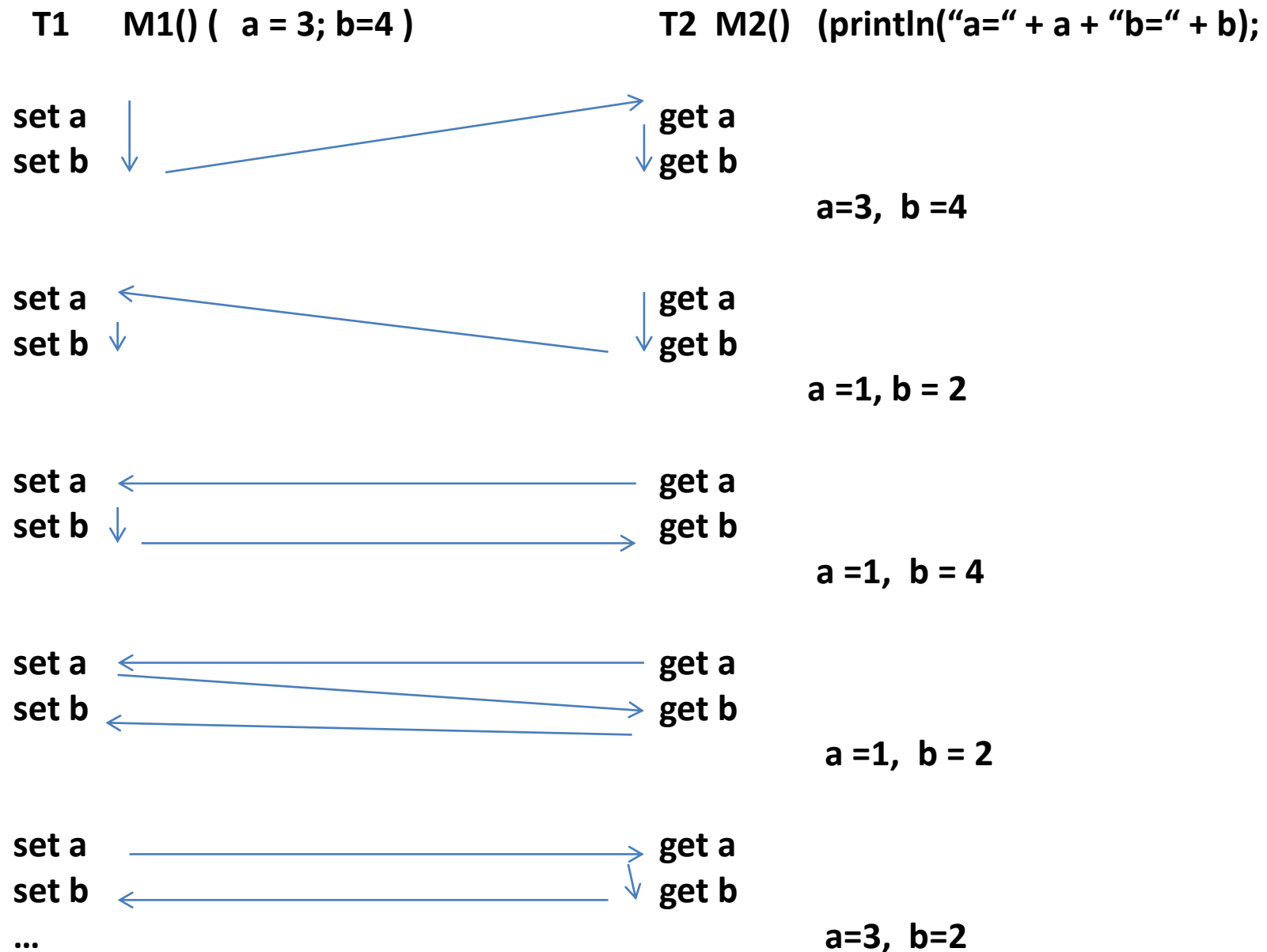
Sequências de execução possíveis: (a=1 b=2)



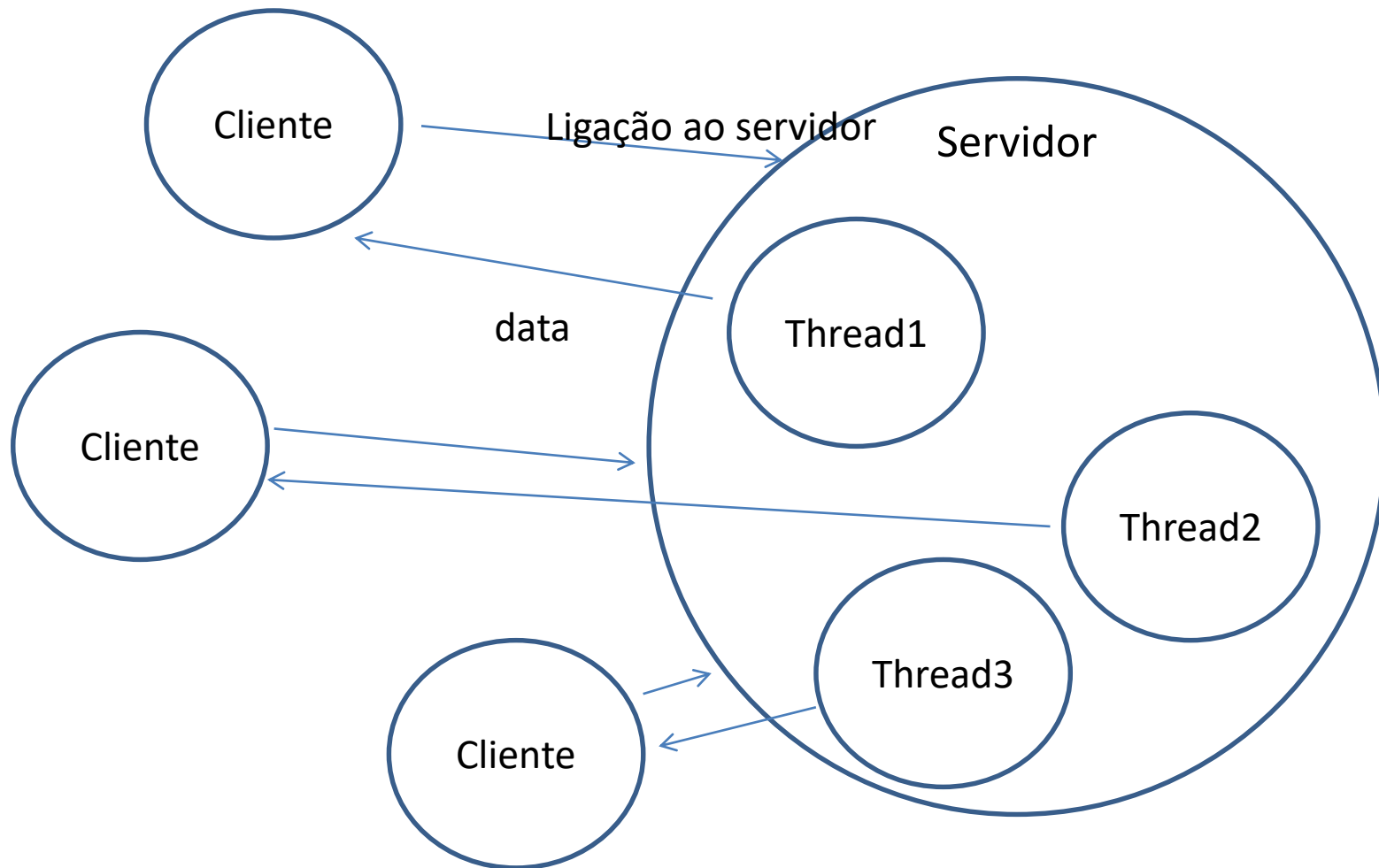
Sincronização de Threads exemplos: (ver T03a)

```
public class Exemplo4 {  
  
    private int a = 1, b = 2;  
  
    public void M1()  
        { a = 3; b = 4; }  
  
    public void M2()  
        { System.out.println ("a=" + a + "b=" + b); }  
  
}
```

Sequências de execução possíveis: (a=1 b=2)



Multithreaded Server (T03a – pág. 32)



Programação concorrente em Java

Exercício 2: Suponha uma aplicação cliente/servidor em que :

- Cada cliente aposta num número inteiro de 0 a 99;
- Servidor gera um número inteiro aleatório entre 0 e 99 e se for igual à aposta do cliente este ganha um determinado prémio. Note que para cada cliente deve ser gerado um novo número.

O valor do prémio é determinado da seguinte forma: é inicializado a 0; para cada cliente que joga é adicionado ao prémio o valor de 1€. Quando um cliente acerta no número gerado ganha 50% do valor acumulado até esse momento no prémio. Após um cliente acertar no prémio, o prémio volta a zero.

- O Servidor devolve uma mensagem com o texto “ Parabéns, ganhou XXX €” caso acerte e em que XXX deve conter o valor ganho. Caso o cliente não acerte no valor gerado a mensagem deverá ser “Continue a tentar, o prémio já é de YYY €” em que YYY é o valor que poderia ganhar nesse momento.

Programação concorrente em Java

– Implemente a aplicação anterior usando Sockets TCP para comunicação entre o cliente e um servidor multithreaded. Use ObjectStreams.

a) Construa o processo cliente.

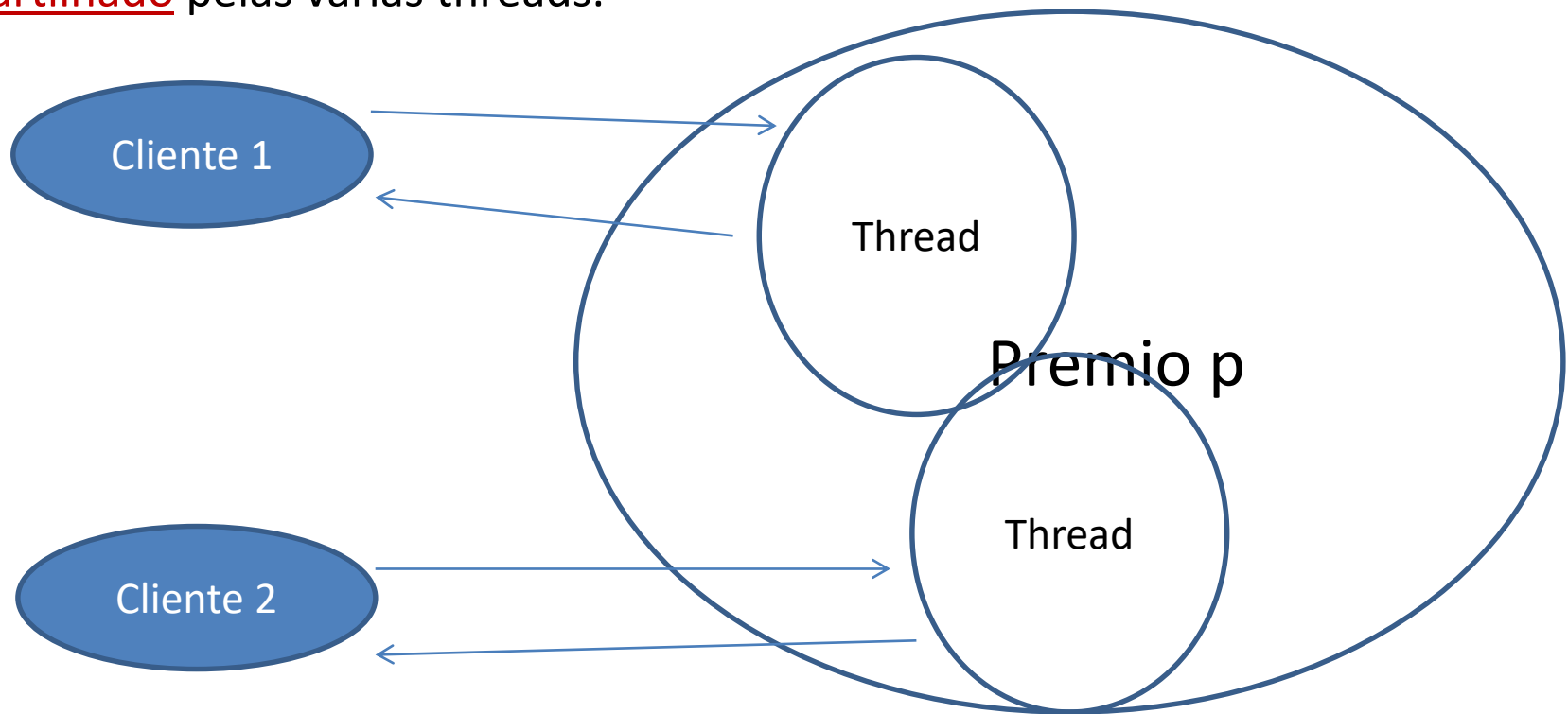
b) Construa o processo servidor

c) Construa a Thread que comunica com o cliente.

Programação concorrente em Java

Começamos por construir uma classe, que vai conter o valor do prémio em cada instante.

Para cada cliente é criada uma thread no servidor. O objeto do tipo Premio é partilhado pelas várias threads.



Programação concorrente em Java

Classe Premio

```
public class Premio {  
    private int valor;  
  
    public int getValor() {  
        return valor;  
    }  
    //Alteramos o valor do prémio em exclusão mútua  
    public synchronized void setValor(int valor) {  
        this.valor = valor;  
    }  
    public synchronized void incValor() {  
        valor ++;  
    }  
}
```

Programação concorrente em Java

Classe Cliente

```
public class Cliente {  
    private Socket s;  
    public Cliente(){  
        int aposta = 0;  
        try {  
            s = new Socket ("127.0.0.1", 5432);  
            ObjectOutputStream os = new ObjectOutputStream  
                                                                    (s.getOutputStream());  
            ObjectInputStream is = new ObjectInputStream( s.getInputStream() );  
  
            System.out.println(" Qual a sua aposta?")  
            aposta = Ler.umInt();  
            os.writeObject(aposta);  
            os.flush();  
            System.out.println(is.readObject());  
            s.close();  
        }  
    }  
}
```

Programação concorrente em Java

Classe Cliente

```
    catch ( IOException e)
        { System.out.println(e.getMessage());}
    catch (ClassNotFoundException ex)
        { System.out.println(ex.getMessage()); }

} //construtor
public static void main (String args []){
    Cliente c = new Cliente();
}
} // fim da classe cliente
```

Programação concorrente em Java

Classe Servidor

```
public class Servidor {  
    private ServerSocket ss;  
    private Socket s;  
    private ThreadServidor c;  
    public Servidor(){  
        Premio p = new Premio();  
        try {  
            ss = new ServerSocket (5432);  
        }  
        catch ( IOException e){  
            System.out.println(e.getMessage());  
        }  
        try {  
            while (true) {  
                s = ss.accept();  
                c = new ThreadServidor ( s, p ); // passamos para a Thread o socket  
                                                    // e o prémio  
            }  
        }  
    }  
}
```

Programação concorrente em Java

Classe Servidor

```
        catch (IOException e) {  
            System.out.println(e.getMessage());  
        }  
    }  
    public static void main (String args[]){  
        Servidor server = new Servidor();  
    }  
} // fim da classe servidor
```

Programação concorrente em Java

Classe da Thread que é criada quando da ligação de cada cliente ao servidor

```
public class ThreadServidor extends Thread {  
    private Socket S;  
    private Premio P;  
    public ThreadServidor ( Socket s, Premio p) {  
        super();  
        S = s;  
        P=p;  
        start();  
    } // fim do construtor  
  
    // public void run () {
```

Programação concorrente em Java

Classe da Thread que é criada quando da ligação de cada cliente ao servidor

```
public void run () {  
    try {  
        ObjectOutputStream os = new ObjectOutputStream (S.getOutputStream());  
        ObjectInputStream is = new ObjectInputStream( S.getInputStream() );  
        int v = (int)is.readObject(); //recebe a aposta  
        P.incValor(); // incrementa o valor do prémio  
        int guess =(int) (Math.random()*100); //gera o valor aleatório  
        double ganhou = P.getValor()/2.0;  
        if (v == guess){  
            P.setValor(0);  
            os.writeObject ("Parabéns Ganhou " + ganhou+ "€" );  
        }else  
            os.writeObject ("Continue a tentar, o prémio já é de " + (2*ganhou)+ "€");  
        os.flush();  
    }  
}
```

Programação concorrente em Java

Classe da Thread que é criada quando da ligação de cada cliente ao servidor

```
public void run () {
```

```
...
```

```
    catch ( IOException e) {  
        System.out.println(e.getMessage());  
    }  
    catch (ClassNotFoundException ee){  
        System.out.println(ee.getMessage());  
    }  
}
```

Leitura de dados do teclado com tratamento de erros:

```
public class Ler {  
  
    public static String umaString () {  
        String s = "";  
        try {  
            BufferedReader in = new BufferedReader ( new InputStreamReader  
(System.in));  
            s = in.readLine();  
        } catch (IOException e) {  
            System.out.println("Erro ao ler fluxo de entrada.");  
        }  
        return s;  
    }  
}
```

```
public static int umInt(){
    while(true){
        try{
            return Integer.valueOf(umaString().trim()).intValue();
        }
        catch(Exception e){
            System.out.println("Não é um inteiro válido!!!");
        }
    }
}

public static byte umByte(){
    while(true){
        try{
            return Byte.valueOf(umaString().trim()).byteValue();
        }
        catch(Exception e){
            System.out.println("Não é um byte válido!!!");
        }
    }
}
```

```
public static short umShort(){
    while(true){
        try{
            return Short.valueOf(umaString().trim()).shortValue();
        }
        catch(Exception e){
            System.out.println("Não é um short válido!!!");
        }
    }
}

public static long umLong(){
    while(true){
        try{
            return Long.valueOf(umaString().trim()).longValue();
        }
        catch(Exception e){
            System.out.println("Não é um long válido!!!");
        }
    }
}
```

```
public static float umFloat(){
    while(true){
        try{
            return Float.valueOf(umaString().trim()).floatValue();
        }
        catch(Exception e){
            System.out.println("Não é um float válido!!!");
        }
    }
}

public static double umDouble(){
    while(true){
        try{
            return Double.valueOf(umaString().trim()).doubleValue();
        }
        catch(Exception e){
            System.out.println("Não é um double válido!!!");
        }
    }
}
```

```
public static char umChar(){  
    while(true){  
        try{  
            return umaString().charAt(0);  
        }  
        catch(Exception e){  
            System.out.println("Não é um char válido!!!");  
        }  
    }  
}
```

```
public static boolean umBoolean(){  
    while(true){  
        try{  
            return Boolean.valueOf(umaString().trim()).booleanValue();  
        }  
        catch(Exception e){  
            System.out.println("Não é um boolean válido!!!");  
        }  
    }  
}  
} // class Ler
```