INSTRUCCIONES IMPORTANTES EXAMEN FEBRERO PSP

```
Scanner scanner=new Scanner(System.in);
int numeroIntroducido=scanner.nextInt();
scanner.nextLine(); // Limpiar el buffer para poder volver a usarlo

Random random = new Random();
int numeroRandom = random.nextInt(5)+1; // numero aleatorio entero entre 1 y 5

List<Thread>hilos=new ArrayList(); // Agregar todos los hilos a una lista.
hilo.start() // lanzar hilo, se ejecuta su método run()
hilo.join() // esperar que acaben todos los hilos.

Productor productor=new Productor("Productor",recursoCompartido); // Productor implements Runnable
Thread hiloProductor=new Thread(productor);

public synchronized void metodoSincronizado
wait(); // el hilo espera que se libere el recurso
notifyAll(); // el hilo notifica al resto que comprueben sus condiciones
panadero.interrupt(); // interrumpir un hilo en ejecución

BufferedReader br usa br.readLine()
PrintWriter pw usa pw.println()
```

```java
// MENU
public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int opcion;

        do {
            System.out.println("\n=== MENÚ ===");
            System.out.println("1. Opción 1");
            System.out.println("2. Salir");
            System.out.print("Seleccione una opción: ");

            opcion = scanner.nextInt();

            switch (opcion) {
                case 1:
                    System.out.println("Has seleccionado la Opción 1.");
                    break;
                case 2:
                    System.out.println("Saliendo del programa...");
                    break;
                default:
                    System.out.println("Opción no válida. Inténtalo de nuevo.");
            }
        } while (opcion != 2);

        scanner.close();
}
```

```java
// SERVIDOR TCP
public class Servidor extends Thread {
        private final Socket skCliente;
        public Servidor(Socket skCliente) {
                this.skCliente = skCliente;
        }
        public static void main(String[] args) {
                try {
                        final int PUERTO = 55555;
                        ServerSocket skServidor = new ServerSocket(PUERTO);
                        System.out.println("Servidor escuchando en el puerto " + PUERTO);
                        while (true) {
                                Socket skCliente = skServidor.accept();
                                new Servidor(skCliente).start();
                        }
                } catch (IOException ex) {
                        System.out.println(ex.getMessage());
                }
        }
        @Override
        public void run() {
           // Gestión de clientes que se conectan
           try {
              BufferedReader br = new BufferedReader(new InputStreamReader(socket.getInputStream()));
              PrintWriter pw = new PrintWriter(socket.getOutputStream(), true);
              pw.println("Texto enviado");
              String textoLeido = br.readLine();
              } catch (IOException ex) {
                    System.out.println(ex.getMessage());
              }
        }
}
```

```
// CLIENTE TCP
try(Socket clienteSocket = new Socket(HOST, PUERTO);
BufferedReader br = new BufferedReader(new InputStreamReader(clienteSocket.getInputStream()));
PrintWriter pw = new PrintWriter(clienteSocket.getOutputStream(), true)) // autoflush activado
{
        // Comunicacion con el servidor
}catch(IOException ex){
        System.out.println(ex.getMessage());
}
```

CAPTURAR Y GESTIONAR EXCEPCIONES try/catch o try/catch with resources

- IOException, InterruptedException, Exception

CERRAR FLUJOS DE ENTRADA/SALIDA Y SOCKETS

- scanner.close();
- socket.close();
- br.close();
- pw.close();

DOCUMENTAR EL CÓDIGO

COLORES EN CONSOLA

- System.out.println("\u001B[33m---- AMARILLO ----\u001B[0m");
- System.out.println("\u001B[32m---- VERDE ------\u001B[0m");
- System.out.println("\033[1;31;47m------ ROJO -------\033[0m");
- System.out.println("\u001B[36m----- AZUL CYAN -----\u001B[0m");