Clase 01/10/2020

Multihilos

```
class P
{
  static class Worker extends Thread  -> Esta es una clase dentro de la clase P que extiende de la clase Thread, por eso se puede usar sus métodos

  {
    public void run()
    {
        //Aqui va lo que se quiera hacer el hilo
    }
  }
  public static void main(String[] args) throws Exception
  {

  }
}
```

Para crear la ejecución de un hilo, se crea una isntancia de la clase que esta dentro en este clase Worker e incoamos el método start el cual es heredado del Thread

```
Worker w = new Worker();
w.start();
```

El meted join sirve para tener un estado de espera pasiva mientras el hilo “w” se esta ejecutando, cuando este termina el método join regresa, se dice que cuando un hilo espera la terminación de uno o mas hilos se implementa una barreara.

```
w.join();
```

Synchronized evita que dos o más threads ejecuten simultáneamente un bloque de instrucciones. Al bloque de instrucciones que solo puede ser ejecutado por un thread se le llama **sección crítica**.

```
synchronized(objeto){
    instrucciones
}
```

Para que un servidor pueda recibir varios clientes se usan multihilos, Dentro de la clase Worker creamos un socket conexión y como paramatero de ella se recibe la conexion que ha llegado y dentro de ella decimos que la conexión que creamos es igual a la conexion que acaba de llegar.
En la clase main del servidor creamos nuestro ServerSocket y un ciclo infinito que estará esperado impere la conexion de un nuevo cliente. Dentro de este vamos a crear un Socket de oxnexion y vamos a aceptarlos con el método accept, creamos el hilo y le pasamos la conexión que acaba de llehar, y posteriormente solo corremos el hilo,

```
class Servidor2
{
  static class Worker extends Thread{
    Socket conexion;
    Worker(Socket conexion)
    {
      this.conexion = conexion;
    }
    public void run()
    {
    }
  }
  public static void main(String[] args) throws Exception
  {
    ServerSocket servidor = new ServerSocket(50000);
    for (;;)
    {
      Socket conexion = servidor.accept();
      Worker w = new Worker(conexion);
      w.start();
    }
  }
}
```

Para que el cliente haga reintento al tratar de conectarse con un servidor por x cuestión, primero creamos un Socket conexión que sea nulo, y dentro de un ciclo infinito le decimos al Socket que intente conectarse con el servidor de forma normal.En caso que no se pueda podemos dormir el hilo para que lo vuelva a integre en un tiempo N después.

```
Socket conexion = null;
for(;;)
```

```
try
{
  conexion = new Socket("localhost",50000);
  break;
}
catch (Exception e)
{
  Thread.sleep(100);
}
```