Clases

Clase – Definición

- Una clase es una construcción que nos permite crear nuestros propios tipos mediante el agrupamiento de variables de otros tipos, métodos y eventos
- Una clase es como una plantilla. Define los datos contenidos y el comportamiento que tendrá nuestro tipo de datos
- Para poder usarlas, uno crea *instancias* a través de las cuales accede a sus propiedades y métodos

Clases

- Sintaxis

```
[modificadores] class NombreClase{}
```

- Las clases se nombran con la primer letra de cada palabra en mayúscula (notación Pascal)
- Suelen utilizarse sustantivos como nombre de clases

Clases - Ejemplo

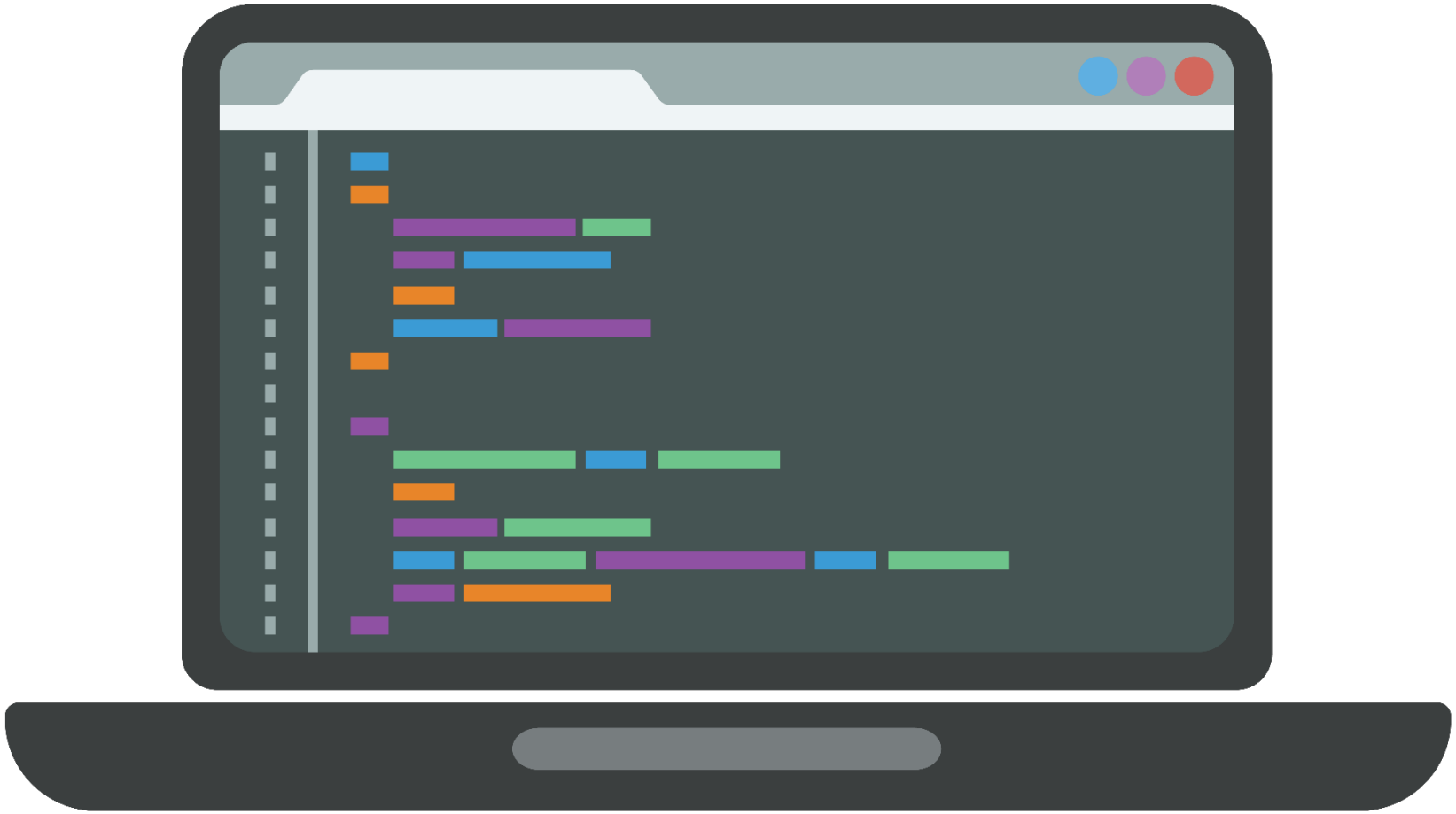
```
public class Alumno{
   private string Nombre;
   private string Apellido;
   private int Edad;

   public DarNombre(string nombre){
      this.Nombre = nombre;
   }
}
```

Clases - Utilización

- En la mayoría de los casos, vamos a trabajar en nuestros programas con una o más instancias de las clases que hayamos definido en nuestro programa

```
NombreClase nombreInstancia = new NombreClase(); // Instancia de la clase
tipo_retorno valorRetorno = nombreInstancia.Metodo(); //Metodo con retorno
nombreInstancia.MetodoVoid();//Metodo sin retorno
tipo_propiedad valorPropiedad = nombreInstancia.Propiedad;//Leo valor propiedad
```

CAPTTON
IT CONSULTING



CAPTTON
IT CONSULTING

Métodos

Métodos - Definición

- Los métodos son porciones de códigos que encapsulan una funcionalidad **concisa**
- Un método es el equivalente a las funciones, procedimientos o subrutinas de otros lenguajes
- En Java los métodos están siempre asociados a una clase

Métodos - Ventajas

- Los métodos permiten que los programas sean mas legibles y fáciles de mantener
- Los métodos permiten que el desarrollo y el mantenimiento de los sistemas sean mas rápidos
- Los métodos son centrales en la reutilización del software
- Los métodos permiten que diferentes objetos se comuniquen y distribuyan el trabajo desarrollado por el programa

Métodos - Sintaxis

```
[modificadores]tipo_retorno NombreMetodo([argumentos]) {
        //Codigo del metodo
    }
```

- `tipo_retorno` – todos los tipos presentes en el lenguaje + ‘void’ cuando no se espera un retorno
- `NombreMetodo` – Se pone en mayúscula cada palabra, usualmente se utilizan verbos (notación Pascal)
- argumentos – pueden ser 0 o más argumentos. Se declara su tipo explícitamente

Métodos - Sintaxis

- Ejemplo

```
void MostrarMenu(){//bloque   de codigo}
int SumarDosNumeros(int numero1,int numero2){
  int resultado = numero1 + numero2;
  return resultado;
}
string[] OrdenarVector(string[] vector){
  string[] ordenado (……)
  return ordenado;
}
```

Métodos – Sobrecarga (Overloading)

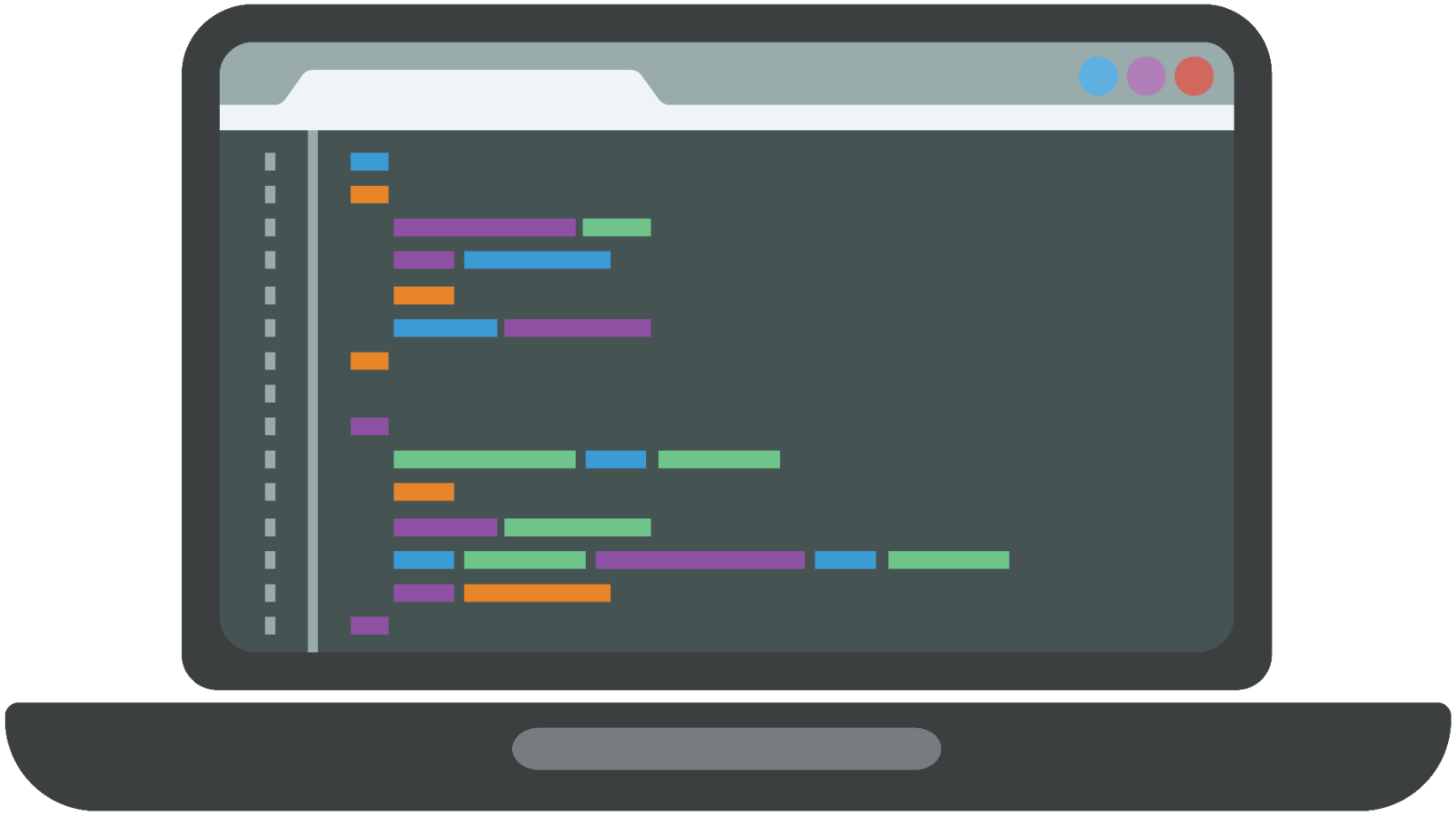
- Hablamos de sobrecarga de un método cuando nos referimos a la posibilidad de que el mismo método acepte distintos tipos o cantidades de parámetros
  - En vez de declarar:

```
int SumarDosNumeros(int numero1, int numero2){}
int SumarTresNumeros(int numero1, int numero2,int numero3){}
float SumarFlotantes(float numero1, float numero2){}
```

  - Declaramos:

```
int Sumar(int numero1, int numero2){}
float Sumar(float numero1, float numero2){}
int Sumar(int numero1, int numero2,int numero3){}
```

CAPTTON
IT CONSULTING



Modificadores

Modificadores de Acceso

- Los modificadores de acceso se utilizan para limitar el acceso a un método o a una clase desde otras clases
- Los modificadores principales son
  - Public
  - Protected
  - Private

Modificador de Acceso Public

- Modificador más permisivo, el acceso no está restricto
- Los métodos o atributos que se utilicen para comunicarse entre clases deben ser de este tipo

Modificador de Acceso Protected

- Modificador de seguridad media
- Dentro de una misma clase o sus hijos, los métodos o atributos protegidos son accesibles
- Fuera de la clase, los métodos o atributos no son accesibles

Modificador de Acceso Private

- Modificador mas restrictivo
- Los métodos o propiedades marcados como Privado no puede ser accedido desde fuera de la clase que los contiene ni siquiera por sus hijos

Modificadores de Acceso - Comparación

Modificador	Clase	Paquete	Subclase	Mundo
Public	**Visible**	**Visible**	**Visible**	**Visible**
Protected	**Visible**	**Visible**	**Visible**	**No Visible**
Sin Modificador	**Visible**	**Visible**	**No Visible**	**No Visible**
Private	**Visible**	**No Visible**	**No Visible**	**No Visible**

Modificador Static

- El modificador static se utiliza para definir propiedades o métodos que son inherentes al tipo de dato definido por nuestra clase y no a sus instancias particulares
- Ejemplo

```
public class Alumno{

   static int CantidadAlumnos; //La cantidad de alumnos no le importa a cada alumno
   //particular. Es relevante para la clase en si misma
   private string Nombre;
   private string Apellido;
   private int Edad;

   public DarNombre(string nombre){
      this.Nombre = nombre;
    }
}
```