1. Constructors should have same name as that of class

2. Construct will be called when ever an object is created

3. Constructors by default it is internally void, which means it can never return any value

4. If we use void before the constructor then that constructor will be treated as a method

EXAMPLE !

5. We can create arguments in constructor

6. We can create any number of constructors in same class provided the constructor has got different

number of arguments or different type of argument

7. If we donot add a constructor in .java file then during compilation automatically .class file will include an empty body construtor and hence during object creation we get no error, because in java when object is created it is mandatory to call a constructor

```
package constructorsexample;


public class A {
	A(){
		System.out.println("FROM Constructor");
	}

	public static void main(String[] args) {
		A a1 = new A();
		A a2 = new A();
	}

}
```

output

FROM Constructor

FROM Constructor

Example 2:

```
package constructorsexample;

public class A {
	A(){
		System.out.println("FROM Constructor");
	}

	public static void main(String[] args) {
		new A();
		new A();
	}

}
```

output

FROM Constructor

FROM Constructor

Example 3:

```
package constructorsexample;

public class A {
	A(){
		return 100;
```

```
	}

	public static void main(String[] args) {
	A a1 =   new A();
	}

}
```

Output:

Error

Example 4:

```
package constructorsexample;
// In java anything that is void would support usage of return keyword
public class A {
	A(){
		System.out.println("From Constructor A");
		return;
	}

	public static void main(String[] args) {
	A a1 =   new A();
	}

}
```

Output:

From Construct A

Example 5:

```
public class A {
	void A(){//Method
		System.out.println("From Constructor A");
		return;
	}

	public static void main(String[] args) {
	A a1 =   new A();
	}

}
```

Output:

Will execute but will print nothing

Example 6:

```
public class A {
	void A() {// Method
		System.out.println("From Constructor A");
		return;
	}

	public static void main(String[] args) {
		A a1 = new A();
		a1.A();
```

```
	}


}
```

Output:

From Constructor A

Example 7:

```
public class A {
	A(int i) {// Method
		System.out.println(i);
	}

	public static void main(String[] args) {
		A a1 = new A(100);
	}

}
```

Output:

100

Example 8:

```
public class A {
	A(int i,int j) {// Method
		System.out.println(i);
		System.out.println(j);
	}
```

```
	public static void main(String[] args) {
		A a1 = new A(100,200);
	}

}
```

Output:

100

200

Example 9:

```
public class A {
	A(){//ZERO arguments
		System.out.println(500);
	}
	A(int i){// One Argument
		System.out.println(i);
	}
	A(int i, int j){//Two arguments
		System.out.println(i);
		System.out.println(j);
	}

	public static void main(String[] args) {
		A a1 = new A();
		A a2 = new A(100);
		A a3 = new A(500,600);
	}
```

```
}
```

Output:

```
500
100
500
600
```

Example 10:

```
package constructorsexample;

//
public class A {
	A(int i){// 1 argument & type is int
		System.out.println(i);
	}
	A(char j){// 1 argument & type is char
		System.out.println(j);
	}
	public static void main(String[] args) {
		A a1 = new A(100);
		A a2 = new A('a');

	}

}
```

Output:

100

a

Example 11: