##### **Builder pattern is used to simplify the process of building a complex object.**

Example:

Department d = Department.builder().setLocation("Here").setManagerName("dawei").setName("df").build();

Using one statement, we can set multiple attributes without creating a constructor. Also the setting of attributes are dynamic and flexible.

Builder pattern needs lots of code besides the original class.

public class Department {// these are the real variables of Department.

private String name;

private String location;

private String managerName;

private Integer id;

public Department(String name, String location, String managerName, Integer id) {

this.name = name;

this.location = location;

this.managerName = managerName;

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getLocation() {

return location;

}

public void setLocation(String location) {

this.location = location;

}

public String getManagerName() {

return managerName;

}

public void setManagerName(String managerName) {

this.managerName = managerName;

}

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public static DepartmentBuilder builder(){

return new DepartmentBuilder();

}// Builder Class - a helper class.// it caches all the variables first and then in the build() method, all variables will be copied to the real object.

public static class DepartmentBuilder{

private String name;

private String location;

private String managerName;

private Integer id;

public DepartmentBuilder setName(String name) {

this.name = name;

return this;

}

public DepartmentBuilder setLocation(String location) {

this.location = location;

return this;

}

public DepartmentBuilder setManagerName(String managerName) {

this.managerName = managerName;

return this;

}

public DepartmentBuilder setId(Integer id) {

this.id = id;

return this;

}

public Department build(){

Department d = new Department(name, location, managerName, id);

return d;

}

}

@Override

public String toString() {

return "Department [name=" + name + ", location=" + location + ", managerName=" + managerName + ", id=" + id

+ "]";

}

}

##### **Differences between factory and builder**

Factory is to hide the creation of the object from the client. Client doesn't know how the objects are created. Just ask for it and use it.

Builder is to simplify the creation of the obejct if there are too many attributes to set. Client knows exactly what to set.