Builder pattern was introduced to solve some of the problems while creating the Objects , when the Object contains a lot of attributes.

Too Many arguments to pass from client program to the Factory class that can be error prone because most of the time, the type of arguments are same and from client side its hard to maintain the order of the argument.

Some of the parameters might be optional but, we are forced to send all the parameters and optional parameters need to send as NULL.

It says that construct a complex object from simple objects using step-by-step approach.

It provides clear separation between the construction and representation of an object.

Untill the object is built completely using build() method , it will be in the intermediatory state of type Builder , only on calling the build() method , it will return the final Object.

Example : java.lang.StringBuilder

public class Student {  
  
 *// required field* private int rollNumber;  
  
 *// required field* private String name;  
  
 *// Required and specific* private String department ;  
  
 *// Required and specific to department* private List<String> subjects;  
  
 *// Optional and null values are accepted if we do not specify* private String passport ;  
  
 public Student(StudentBuilder studentBuilder){  
 this.rollNumber = studentBuilder.rollNumber ;  
 this.name = studentBuilder.name ;  
 this.department = studentBuilder.department ;  
 this.subjects = studentBuilder.subjects ;  
 this.passport = studentBuilder.passport ;  
 }  
  
 @Override  
 public String toString() {  
 return "Student{" +  
 "rollNumber=" + rollNumber +  
 ", name='" + name + '\'' +  
 ", department='" + department + '\'' +  
 ", subjects=" + subjects +  
 ", passport='" + passport + '\'' +  
 '}';  
 }  
}

public abstract class StudentBuilder {  
  
 final int rollNumber;  
  
 final String name;  
  
 String department ;  
  
 List<String> subjects;  
  
 String passport ;  
  
 *// Make the required fields as final so that we could force initialize them using constructor* protected StudentBuilder(int rollNumber, String name) {  
 this.rollNumber = rollNumber;  
 this.name = name;  
 }  
  
 public abstract StudentBuilder setPassport(String passport) ;  
  
 public abstract Student build() ;  
  
  
}

public class CSEStudentBuilder extends StudentBuilder{  
  
 public CSEStudentBuilder(int rollNumber, String name) {  
  
 super(rollNumber, name);  
 this.department = "CSE" ;  
 List<String> subs = new ArrayList<>();  
 subs.add("DSA");  
 subs.add("Java");  
 subs.add("Python");  
 this.subjects = subs;  
  
 }  
  
 @Override  
 public StudentBuilder setPassport(String passport) {  
 this.passport = passport ;  
 return this ;  
 }  
  
 @Override  
 public Student build() {  
 return new Student(this);  
 }  
  
  
}

public class ECEStudentBuilder extends StudentBuilder{  
 protected ECEStudentBuilder(int rollNumber, String name) {  
  
 super(rollNumber, name);  
 this.department = "ECE" ;  
 List<String> subs = new ArrayList<>();  
 subs.add("Embedded");  
 subs.add("Java");  
 subs.add("DLD");  
 this.subjects = subs;  
  
 }  
  
 @Override  
 public StudentBuilder setPassport(String passport) {  
 this.passport = passport ;  
 return this ;  
 }  
  
 @Override  
 public Student build() {  
 return new Student(this);  
 }  
}

// Client

public class BuilderDesignPattern {  
  
 public static void main(String[] args) {  
 Student cseStudent =  
 new CSEStudentBuilder(120 , "Seshrao" ).setPassport("KLWW112").build();  
 Student eceStudent =  
 new ECEStudentBuilder( 121 , "Pankaj").build() ;  
  
 System.*out*.println(cseStudent);  
  
 System.*out*.println(eceStudent);  
  
 }  
  
}

In the above example , we should be able to create the instances of the student of CSE , ECE easily without specifying the Subjects and Department which are already known.

Output :

Student{rollNumber=120, name='Seshrao', department='CSE', subjects=[DSA, Java, Python], passport='KLWW112'}

Student{rollNumber=121, name='Pankaj', department='ECE', subjects=[Embedded, Java, DLD], passport='null'}

Process finished with exit code 0