Question 3

A university has students taking ICT, Law and Business courses. In the Faculty of ICT, there are Certificate, Diploma and a Degree courses (BSC-IT). BSC-IT has three stages: Stage 1, Stage 2 and Stage 3.

(i) Briefly explain how inheritance can assist in the capture and processing of student details in the above scenario.

Inheritance is a fundamental concept in object-oriented programming that enables a class to inherit properties and behaviors from a parent class. In the context of the given scenario, inheritance can be used to capture and process student details efficiently.

Firstly, a base class called "Student" can be created, which will contain common attributes and methods that are shared among all types of students, such as name, age, contact details, and academic records. This class can be used as a blueprint for all students, regardless of their field of study.

Next, subclasses can be created for each field of study - ICT, Law, and Business. These subclasses will inherit the properties and methods of the Student class and can also have additional fields specific to each field of study. For example, the ICT subclass may have fields such as programming languages, software development tools, and project management skills, while the Law subclass may have fields such as legal specializations, courtroom skills, and legal research methods.

Furthermore, for the BSC-IT degree program, subclasses can be created for each stage of the program - Stage 1, Stage 2, and Stage 3 - which will inherit the properties and methods of the ICT subclass. These subclasses can also have additional fields specific to each stage of the program, such as the courses taken, grades obtained, and project work completed.

Using inheritance in this way allows for the efficient capture and processing of student details, as the common attributes and methods of all students are defined in the base class and the unique attributes and methods of each field of study and each stage of the BSC-IT program are defined in the appropriate subclasses. This approach also promotes code reusability and simplifies the maintenance and updating of the codebase.