* **Types of errors -**

There are basically three types of errors that you must contend with when writing computer programs:

* Syntax errors
* Runtime errors
* Logic errors

**Syntax errors**

In effect, syntax errors represent *grammar errors* in the use of the programming language. Common examples are:

* Misspelled variable and function names
* Missing semicolons
* Improperly matches parentheses, square brackets, and curly braces
* Incorrect format in selection and loop statements

**Runtime errors**

Runtime errors occur when a program with no syntax errors asks the computer to do something that the computer is unable to reliably do. Common examples are:

* Trying to divide by a variable that contains a value of zero
* Trying to open a file that doesn't exist

There is no way for the compiler to know about these kinds of errors when the program is compiled.

**Logic errors**

Logic errors occur when there is a design flaw in your program. Common examples are:

* Multiplying when you should be dividing
* Adding when you should be subtracting
* Opening and using data from the wrong file
* Displaying the wrong message
* AWT and Swing –

|  |  |
| --- | --- |
| **Java AWT** | **Java Swing** |
| AWT components are **platform-dependent**. | Java swing components are **platform-independent**. |
| AWT components are **heavyweight**. | Swing components are **lightweight**. |
| AWT **doesn't support pluggable look and feel**. | Swing **supports pluggable look and feel**. |
| AWT provides **less components** than Swing. | Swing provides **more powerful components**such as tables, lists, scrollpanes, colorchooser, tabbedpane etc. |
| AWT **doesn't follows MVC**(Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view. | Swing **follows MVC**. |

* Concrete Derived Class

A concrete class is a class that has an implementation for all of its methods that were inherited from abstract or implemented via interfaces. It also does not define any abstract methods of its own. This means that an instance of the class can be created/allocated with the **new** keyword without having to implement any methods first. Therefore it can be inferred that any class that is not an abstract class or interface is a concrete class.