One approach popular for requirements analysis is Use Case analysis. Integrated development environments (IDEs) aim to integrate all such help. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. FORTRAN, the first widely used high-level language to have a functional implementation, came out in 1957, and many other languages were soon developed—in particular, COBOL aimed at commercial data processing, and Lisp for computer research. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code. Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages. Techniques like Code refactoring can enhance readability. Computer programmers are those who write computer software. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. As early as the 9th century, a programmable music sequencer was invented by the Persian Banu Musa brothers, who described an automated mechanical flute player in the Book of Ingenious Devices. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Scripting and breakpointing is also part of this process. Techniques like Code refactoring can enhance readability. Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. Different programming languages support different styles of programming (called programming paradigms). Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly. Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute. Integrated development environments (IDEs) aim to integrate all such help. However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. Debugging is often done with IDEs. Standalone debuggers like GDB are also used, and these often provide less of a visual environment, usually using a command line. Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages.