New languages are generally designed around the syntax of a prior language with new functionality added, (for example C++ adds object-orientation to C, and Java adds memory management and bytecode to C++, but as a result, loses efficiency and the ability for low-level manipulation). These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it easy to target varying machine instruction sets via compilation declarations and heuristics. Code-breaking algorithms have also existed for centuries. 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. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Programmable devices have existed for centuries. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. 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. Different programming languages support different styles of programming (called programming paradigms). It affects the aspects of quality above, including portability, usability and most importantly maintainability. It is usually easier to code in "high-level" languages than in "low-level" ones. The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. One approach popular for requirements analysis is Use Case analysis. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Following a consistent programming style often helps readability. 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. For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input. Techniques like Code refactoring can enhance readability. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.