3 TYPES OF PROGRAMMING ERRORS

1. **Compilation errors** – spelling, capitalization, grammar and punctuation errors based on the rules for the particular programming language, Java. These occur when you build (compile) a program.

2. **Run-time errors** – bugs that show up when running the program where there’s either an abnormal ending of the run (the program “bombs” after running only part-way through) or the program keeps running and never stops.

3. **Logic errors** – a different kind of bug that occurs when running the program - the program executes to a normal ending, but produces incorrect output due to a logic problem in the program code.

RUN-TIME ERRORS

These kinds of errors are fairly recognizable – either there’s a system error message in the output (Run-Time) window when you run the program or the program just keeps running because it’s in an infinite loop. The examples on the website let you experiment with these types of problems to see some typical causes of these kinds of bugs – e.g., users providing the wrong type of data as input or providing data that results in dividing by 0.

It’s very important that programmers fully test their program for all types of situations and include additional code to verify and possibly correct any bad user-input. This leads to several programming “rules”:

- Never trust human input as being correct.
- Never trust humans to follow directions in terms of entering input.
- A programmer has to think of EVERYTHING that could go wrong, and have the program deal with it.
- A program may not be able to figure out what to do the bad data or may encounter an irrecoverable error situation. But at least it should
  - have a graceful exit
  - with an error message indicating an error happened
  - and tell the user how to fix the problem, ideally.

LOGIC ERRORS

These are the hardest kinds of errors to find since there are no error messages provided and not necessarily any “obvious symptoms” initially, as there were for run-time errors. These kinds of problems are only detectable when the programmer carefully examines the output to see if it’s fully correct. This is where a good deal of the programmer’s development time is spent – in testing and debugging their program, verifying that all the output is precisely and totally correct. Testing has to include trying every type of input situation when running the program. Then when a problem is found in the output, debugging starts, with the programmer trying to determine why the results are wrong and what line(s) of the code caused the problem. There are many possible causes of logic errors from a programmer not fully understand what the program is supposed to do (according to the client’s requirements), to an accidental leaving off of some crucial step, or putting steps in the wrong order, or not fully testing the program with all possible types of input, etc.

Through practice and experience, programmers learn to reduce the number of such problems (and thus significantly reduce debugging time) they write into the code in the first place by such things as:

- using good program design techniques (including development of and a manual walk-through of pseudocode algorithms);
- using incremental development of a program (always keeping the program in a provably working state);
- designing and writing modular programs, including the use of object-oriented programming;
- using proper structured programming rules;
- using good programming format and style;
- using good naming of variables and methods.