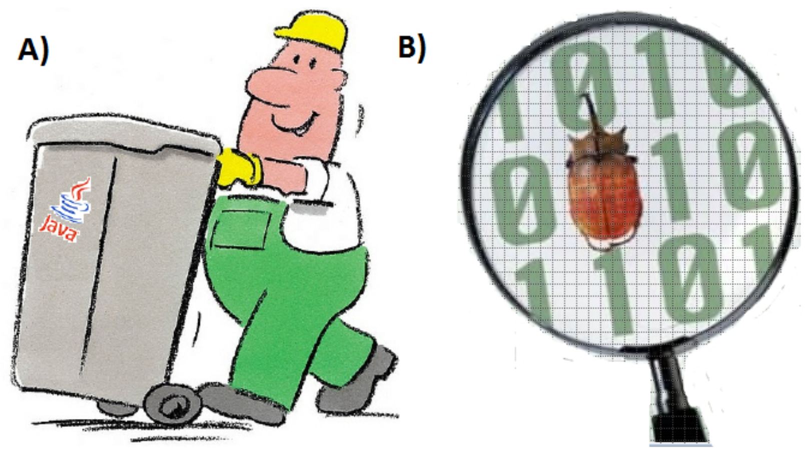

Figure 2.15.: Metaphors in computer science: A) Garbage Collection and B) Debugging

again and discuss the feedback and ideas. The next step is to create a computer-based prototype, where the look and feel is closer to the final product. Nevertheless, the functionality is not an issue during this time. The processes of testing and sharing feedback are repeated and software prototypes are obtained as a result.

According to the four methods previously described, paper prototyping offers benefits regarding the communication within the development team and the quality of the product to be developed. Paper prototypes assure the overall quality of software since they can serve as visual specification of the graphical user interface (GUI). Also, paper prototyping allows communication between the team members regarding the complete design of the required UI. Testing prototypes promote early identification of usability problems even before any code is written, which consequently reduces annoyances of later changes. Finally, but not less important, the costs can be reduced.

2.6. Metaphor in Computer Science

Metaphor is one important writing mechanism which establishes a total identity between two concepts or thoughts without binding them and it is used as symbolic connector. Merriam-Webster's Dictionary defines the word as:

> **met·a·phor** *noun*
>
> *1 : a* figure of speech *in which a word or phrase literally denoting one kind of object or idea is used in place of another to suggest a likeness or* analogy *between them (as in drowning in money); broadly : figurative language - compare simile*

A pair of examples from computer science are shown in Figure 2.15. Garbage Collection (GC) is depicted in Figure 2.15 (A). GC is a special case of resource management in which the limited resource being managed is memory. The garbage collector attempts to reclaim garbage, or memory occupied by objects that are no longer in use by the program. Debugging, depicted in Figure 2.15 (B), is a process of finding and reducing the number of bugs, or defects, in a computer program, thus making it behave as expected.