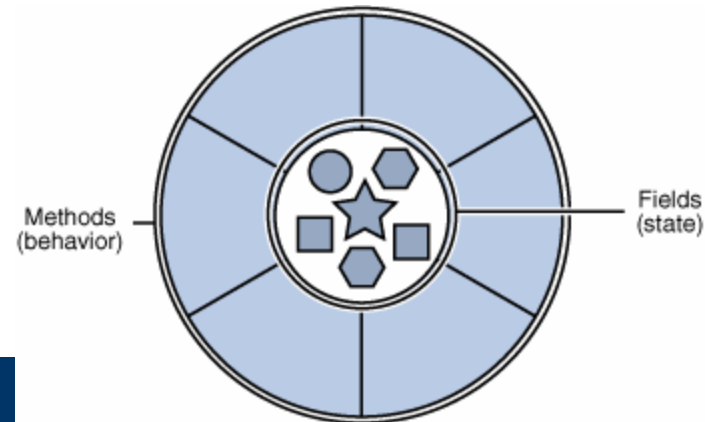


Object Oriented Programming and its relevance to Software Reuse

By Ryan Gerard
Innovim / NASA
GSFC



What are Objects?

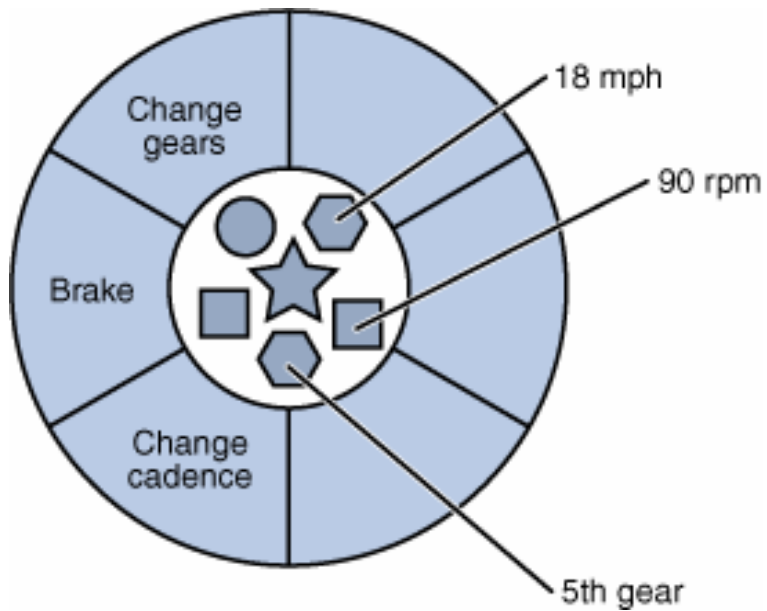


- Software objects are conceptually similar to real-world objects: they too consist of
 - state
 - related behavior
- An object stores its state in *fields* (variables in some programming languages) and exposes its behavior through *methods* (functions in some programming languages).
- Methods operate on an object's internal state and serve as the primary mechanism for object-to-object communication.

Properties of Object Oriented Programming

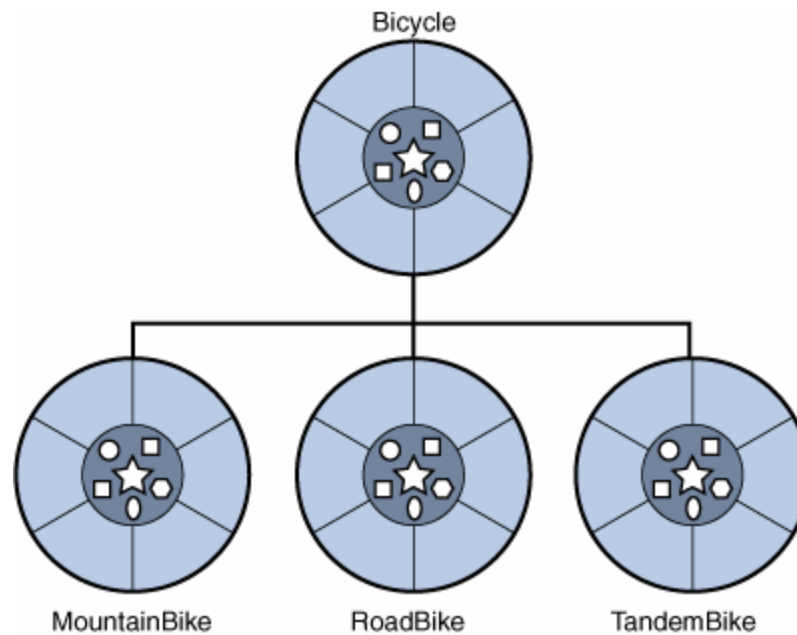
- Modularity: The source code for an object can be written and maintained independently of the source code for other objects.
- Information-hiding: By interacting only with an object's methods, the details of its internal implementation remain hidden from the outside world.
- Software Reuse: If an object already exists (perhaps written by another software developer), you can use that object in your program. This allows specialists to implement/test/debug complex, task-specific objects, which you can then trust to run in your own code.
- Pluggability and debugging ease: “If a bolt breaks, you replace *it*, not the entire machine. ”

Example: A bicycle object



Inheritance

- Different kinds of objects often have a certain amount in common with each other.



Implications to Reuse

- Existing objects can be reused entirely rather than developed from scratch.
- For more specialized objects, complex objects can be implemented by inheriting from simpler objects (these most likely are already available).