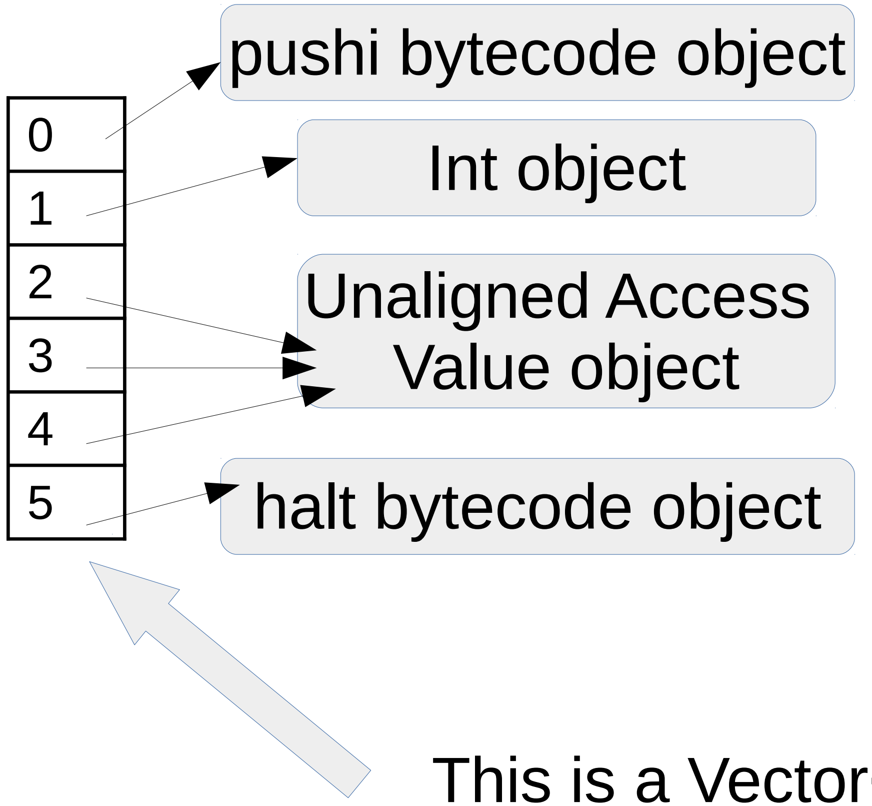
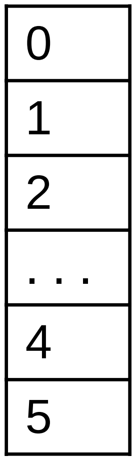
How I’d write the project

Memory is an vector of pointers or references to MemoryObjects

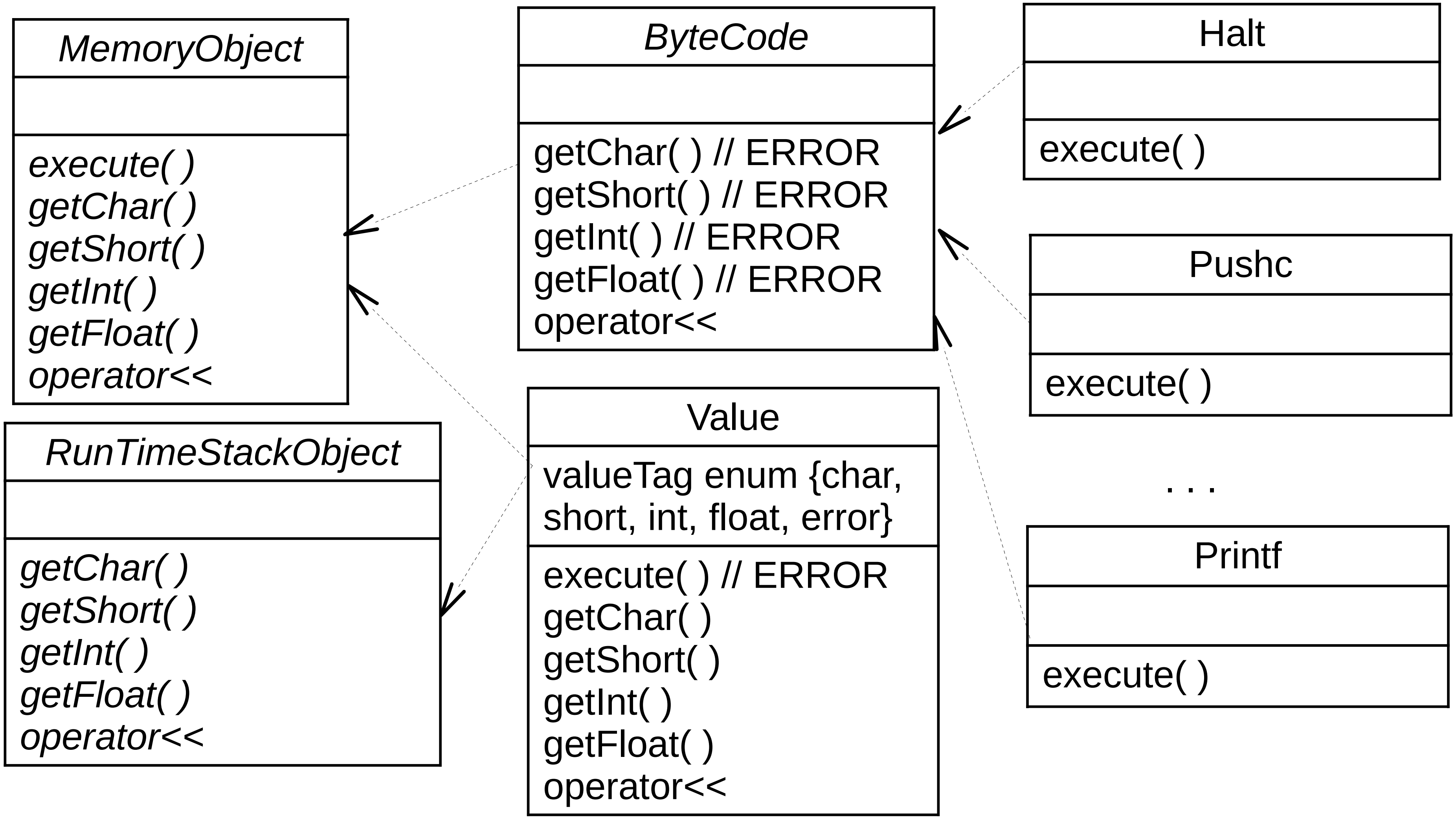
The runtime stack is a vector of pointer or references to RunTimeStackObjects





This is a Vector<MemoryObjects>

Memory is an array of pointers or references



Value objects

- Create one for each value in the memory
- Can also use for runtime stack entries
- Error objects are used to detect unaligned accesses
- execute( ) prints error and dumps the state of the interpreter – have a HASA relationship with the object that contains the interpreter state (stacks, PC, memory)
- For a value of type T, all getX( ), where X not equal to T, print an error and dump the interpreter state.
  - For example, for int, all gets except getInt( ) print error and dump the interpreter state
- Value objects can also be used for RuntimeStack entries
  - for variable offsets just use integer values

ByteCode objects

- There’s a class for every bytecode operation
- Execute performs the function of that bytecode
- The getX( ) methods will give an error and dump the interpreter stack
  - Doing this, and the errors in Value objects will ensure you interpreter gives an error as soon as it tries to access the wrong thing in a memory location

Major functions

1. Initialize the interpreter – create stacks, pointers, initialize pc
2. Initialize, read the program and create MemoryObjects

to run the program:

```
bool continue = true;
while (continue) {
   pc = Memory.getBytecode(pc).execute(pc);
}
```

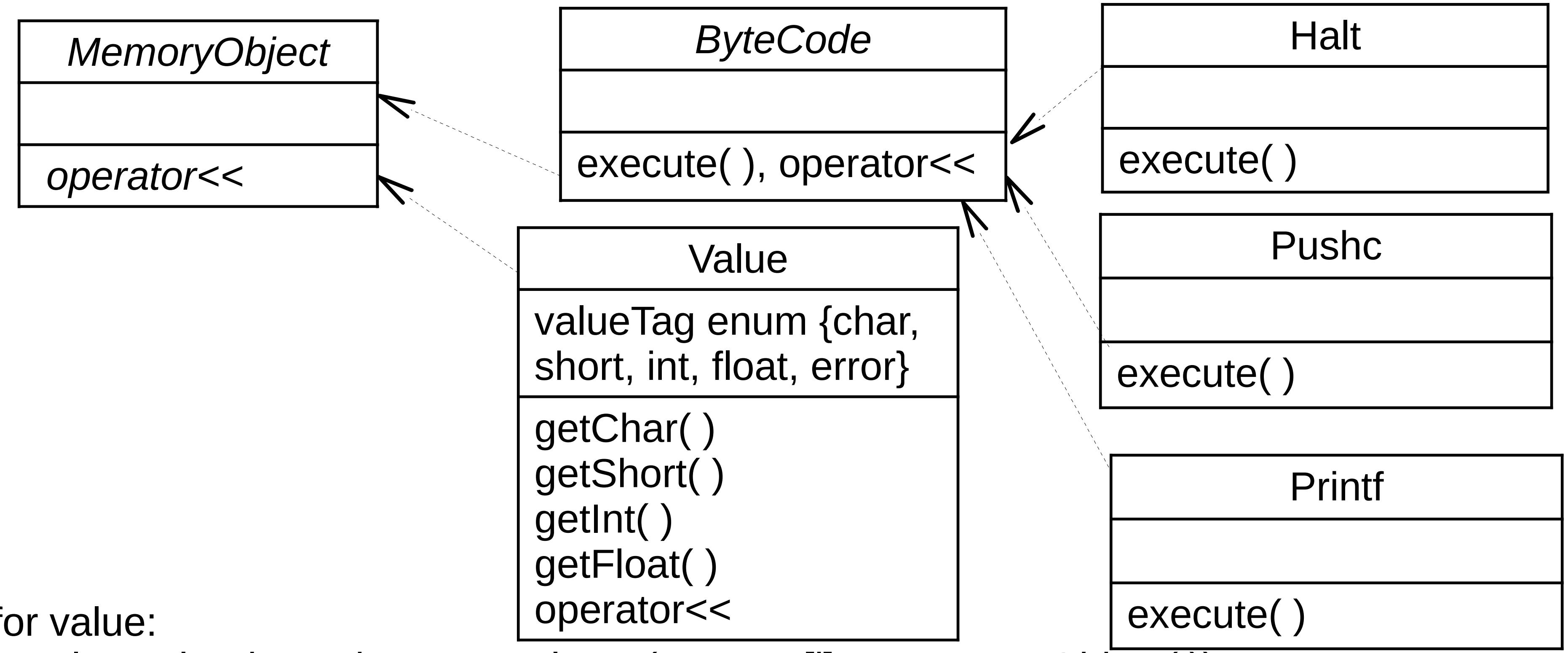
Some advantages of this approach

- Classes largely have a single responsibility
  - This is good, fewer reasons they need to change if the spec changes
  - New opcodes and values easily added without changing other code
    - E.g., could add a long, and everything would work. Only need to change opcodes directly related to this
  - Easy to split work among partners with minimal communication

Remaining ugliness

- I’m not happy with the double inheritance for Value objects
  - Could just use the Memory stuff in the stack, since execute gives an error
- Not happy with getX( ) giving errors for bytecodes and execute( ) giving errors for values
  - Could have what is on the following page
- Probably other imperfections

Memory is an array of pointers or references



```
for value:
   Value val = dynamic_cast<Value*>(Memory[i].getMemoryObject( ));
   if (val == null) // ERROR
for opcode:
   ByteCode bc = dynamic_cast<Bytecode*>(Memory[i].getMemoryObject( ));
   if (bc == null) // ERROR
```