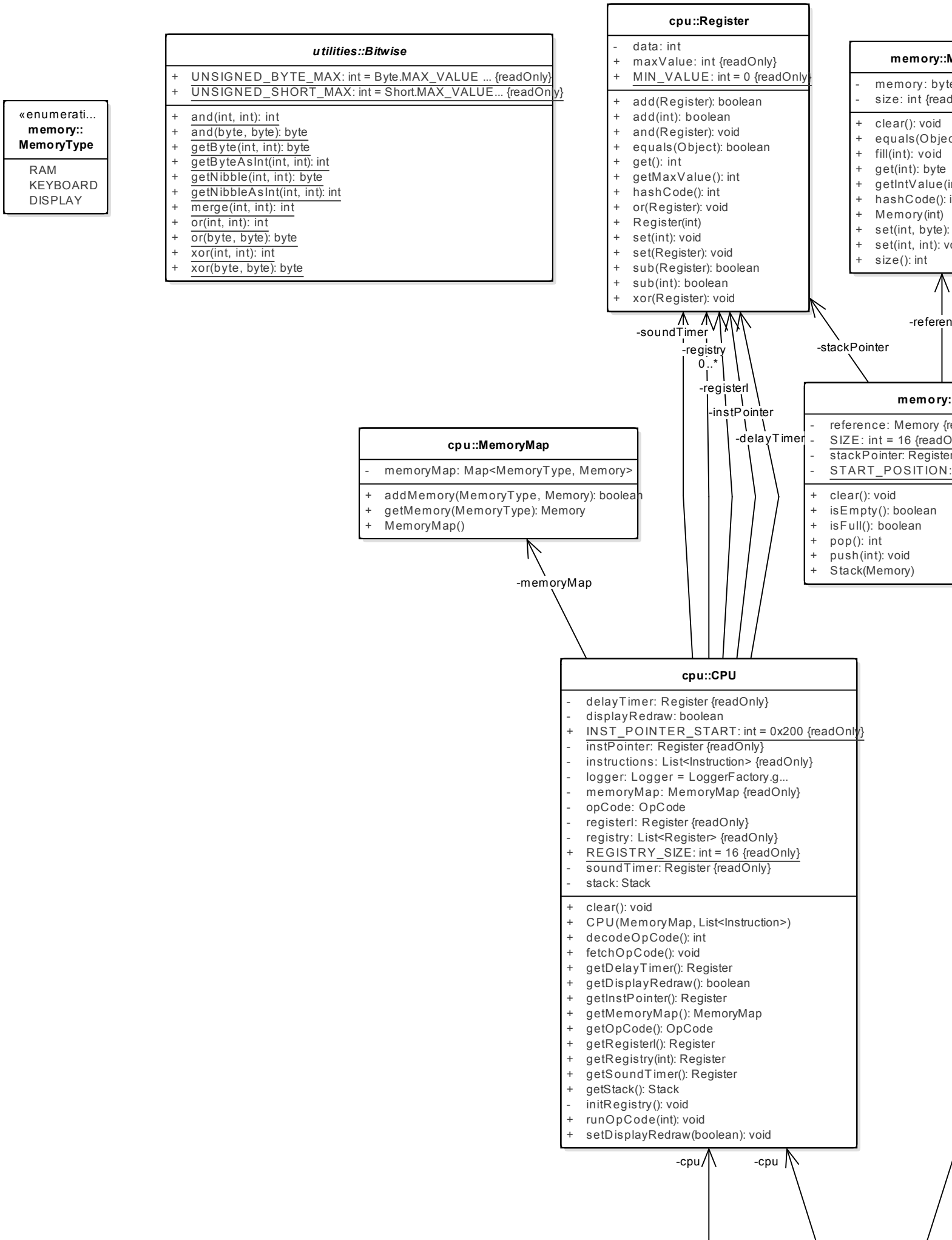
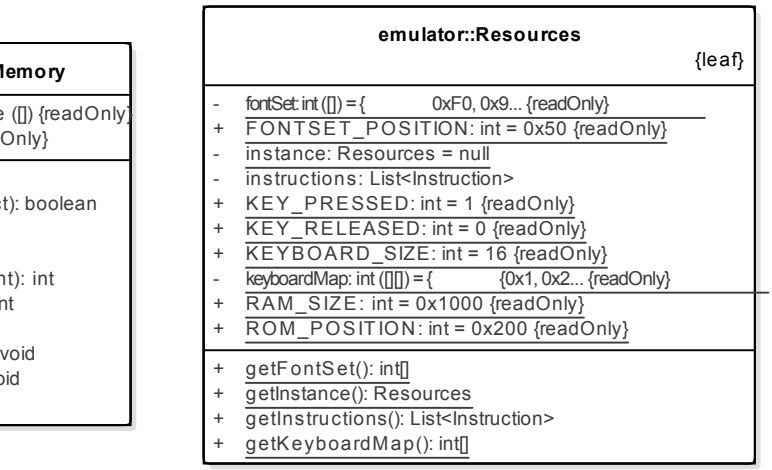
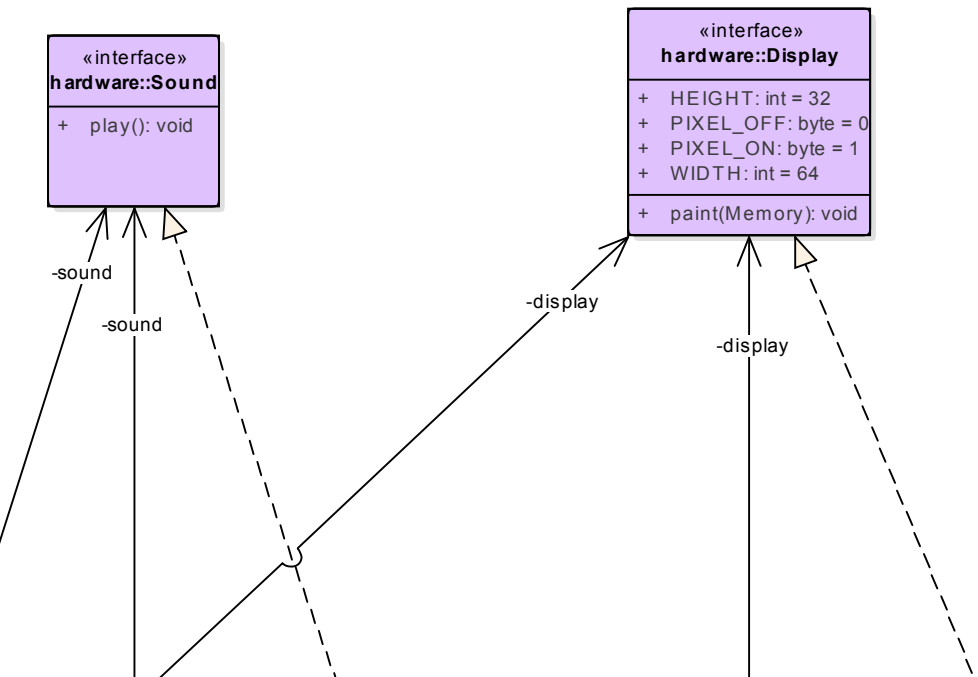
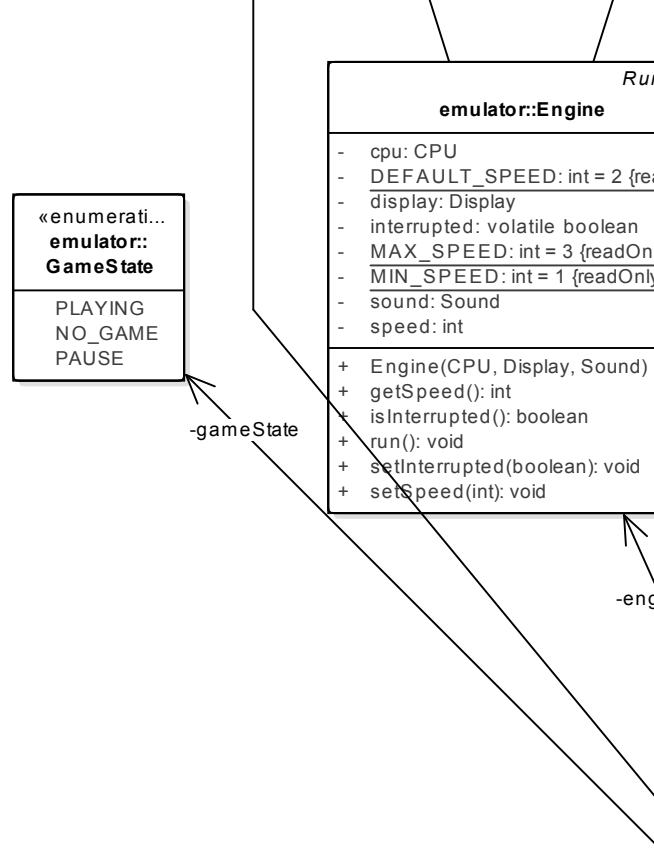
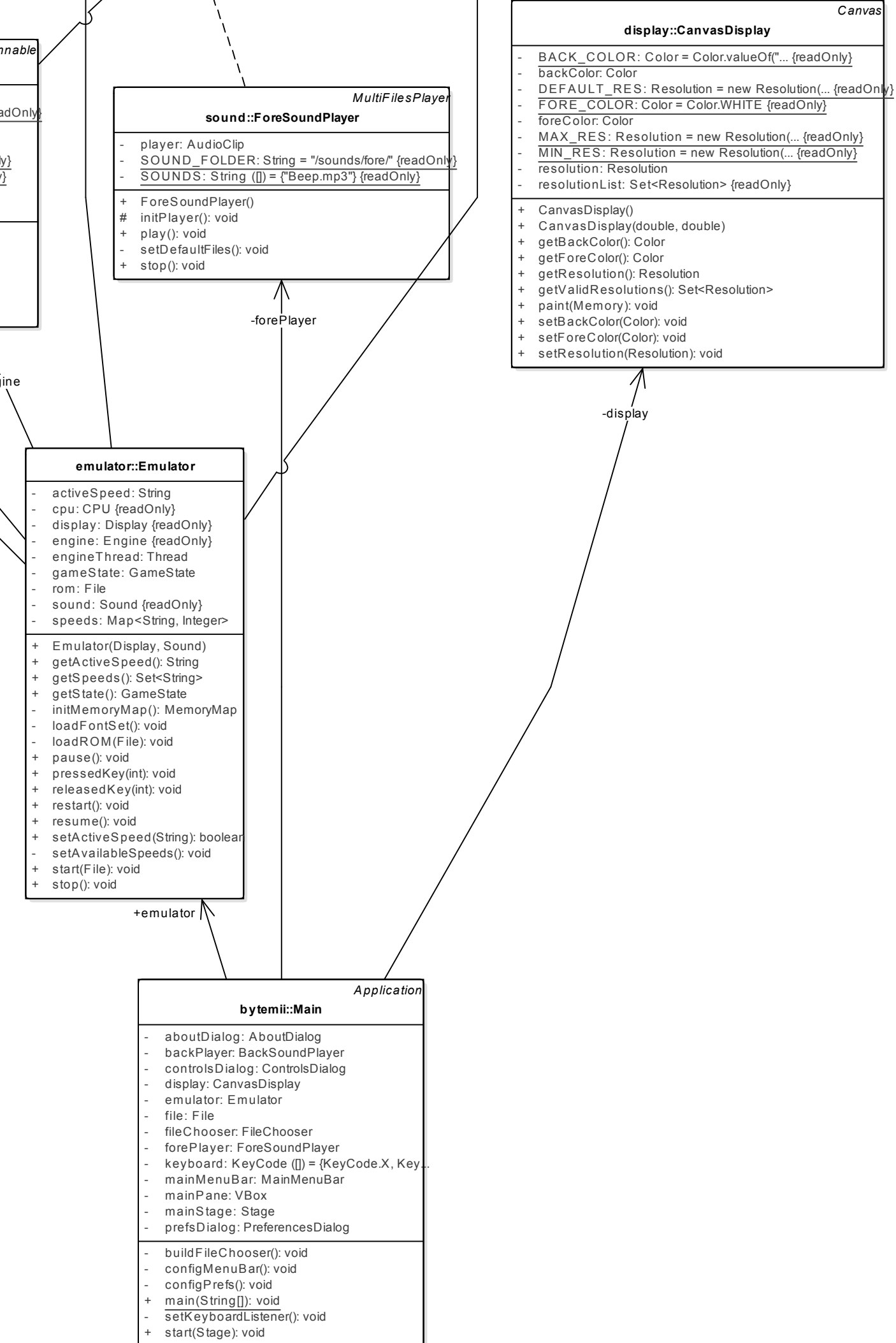

«enumerati...
memory::
MemoryType
RAM
KEYBOARD
DISPLAY
utilities::Bitwise
+ UNSIGNED_BYTE_MAX: int = Byte.MAX_VALUE ... {readOnly}
+ UNSIGNED_SHORT_MAX: int = Short.MAX_VALUE... {readOnly}
+ and(int, int): int
+ and(byte, byte): byte
+ getByte(int, int): byte
+ getByteAsInt(int, int): int
+ getNibble(int, int): byte
+ getNibbleAsInt(int, int): int
+ merge(int, int): int
+ or(int, int): int
+ or(byte, byte): byte
+ xor(int, int): int
+ xor(byte, byte): byte
cpu::Register
- data: int
+ maxValue: int {readOnly}
+ MIN_VALUE: int = 0 {readOnly}
+ add(Register): boolean
+ add(int): boolean
+ and(Register): void
+ equals(Object): boolean
+ get(): int
+ getMaxValue(): int
+ hashCode(): int
+ or(Register): void
+ Register(int)
+ set(int): void
+ set(Register): void
+ sub(Register): boolean
+ sub(int): boolean
+ xor(Register): void
memory::M
- memory: byte
- size: int {read
+ clear(): void
+ equals(Objec
+ fill(int): void
+ get(int): byte
+ getIntValue(i
+ hashCode(): i
+ Memory(int)
+ set(int, byte):
+ set(int, int): vo
+ size(): int
-soundTimer
-registry
0..*
-registerI
-instPointer
-delayTimer
-stackPointer
-referen
cpu::MemoryMap
- memoryMap: Map<MemoryType, Memory>
+ addMemory(MemoryType, Memory): boolean
+ getMemory(MemoryType): Memory
+ MemoryMap()
memory:
- reference: Memory {r
- SIZE: int = 16 {readO
- stackPointer: Register
- START_POSITION:
+ clear(): void
+ isEmpty(): boolean
+ isFull(): boolean
+ pop(): int
+ push(int): void
+ Stack(Memory)
-memoryMap
cpu::CPU
- delayTimer: Register {readOnly}
- displayRedraw: boolean
+ INST_POINTER_START: int = 0x200 {readOnly}
- instPointer: Register {readOnly}
- instructions: List<Instruction> {readOnly}
- logger: Logger = LoggerFactory.g...
- memoryMap: MemoryMap {readOnly}
- opCode: OpCode
- registerI: Register {readOnly}
- registry: List<Register> {readOnly}
+ REGISTRY_SIZE: int = 16 {readOnly}
- soundTimer: Register {readOnly}
- stack: Stack
+ clear(): void
+ CPU(MemoryMap, List<Instruction>)
+ decodeOpCode(): int
+ fetchOpCode(): void
+ getDelayTimer(): Register
+ getDisplayRedraw(): boolean
+ getInstPointer(): Register
+ getMemoryMap(): MemoryMap
+ getOpCode(): OpCode
+ getRegisterI(): Register
+ getRegistry(int): Register
+ getSoundTimer(): Register
+ getStack(): Stack
- initRegistry(): void
+ runOpCode(int): void
+ setDisplayRedraw(boolean): void
-cpu
-cpu


emulator::Resources
{leaf}
- fontSet: int ([]) = { 0xF0, 0x9... {readOnly}
+ FONTSET_POSITION: int = 0x50 {readOnly}
- instance: Resources = null
- instructions: List<Instruction>
+ KEY_PRESSED: int = 1 {readOnly}
+ KEY_RELEASED: int = 0 {readOnly}
+ KEYBOARD_SIZE: int = 16 {readOnly}
- keyboardMap: int ([][]) = { {0x1, 0x2... {readOnly}
+ RAM_SIZE: int = 0x1000 {readOnly}
+ ROM_POSITION: int = 0x200 {readOnly}
+ getFontSet(): int[]
+ getInstance(): Resources
+ getInstructions(): List<Instruction>
+ getKeyboardMap(): int[]


«interface»
hardware::Sound
+ play(): void
«interface»
hardware::Display
+ HEIGHT: int = 32
+ PIXEL_OFF: byte = 0
+ PIXEL_ON: byte = 1
+ WIDTH: int = 64
+ paint(Memory): void
-sound
-sound
-display
-display


Ru
emulator::Engine
- cpu: CPU
- DEFAULT_SPEED: int = 2 {re
- display: Display
- interrupted: volatile boolean
- MAX_SPEED: int = 3 {readOn
- MIN_SPEED: int = 1 {readOnl
- sound: Sound
- speed: int
+ Engine(CPU, Display, Sound)
+ getSpeed(): int
+ isInterrupted(): boolean
+ run(): void
+ setInterrupted(boolean): void
+ setSpeed(int): void
«enumerati...
emulator::
GameState
PLAYING
NO_GAME
PAUSE
-gameState
-eng


MultiFilesPlayer
sound::ForeSoundPlayer
- player: AudioClip
- SOUND_FOLDER: String = "/sounds/fore/" {readOnly}
- SOUNDS: String ([]) = {"Beep.mp3"} {readOnly}
+ ForeSoundPlayer()
# initPlayer(): void
+ play(): void
- setDefaultFiles(): void
+ stop(): void
-forePlayer
Canvas
display::CanvasDisplay
- BACK_COLOR: Color = Color.valueOf("... {readOnly}
- backColor: Color
- DEFAULT_RES: Resolution = new Resolution(... {readOnly}
- FORE_COLOR: Color = Color.WHITE {readOnly}
- foreColor: Color
- MAX_RES: Resolution = new Resolution(... {readOnly}
- MIN_RES: Resolution = new Resolution(... {readOnly}
- resolution: Resolution
- resolutionList: Set<Resolution> {readOnly}
+ CanvasDisplay()
+ CanvasDisplay(double, double)
+ getBackColor(): Color
+ getForeColor(): Color
+ getResolution(): Resolution
+ getValidResolutions(): Set<Resolution>
+ paint(Memory): void
+ setBackColor(Color): void
+ setForeColor(Color): void
+ setResolution(Resolution): void
-display
emulator::Emulator
- activeSpeed: String
- cpu: CPU {readOnly}
- display: Display {readOnly}
- engine: Engine {readOnly}
- engineThread: Thread
- gameState: GameState
- rom: File
- sound: Sound {readOnly}
- speeds: Map<String, Integer>
+ Emulator(Display, Sound)
+ getActiveSpeed(): String
+ getSpeeds(): Set<String>
+ getState(): GameState
- initMemoryMap(): MemoryMap
- loadFontSet(): void
- loadROM(File): void
+ pause(): void
+ pressedKey(int): void
+ releasedKey(int): void
+ restart(): void
+ resume(): void
+ setActiveSpeed(String): boolean
- setAvailableSpeeds(): void
+ start(File): void
+ stop(): void
+emulator
Application
bytemii::Main
- aboutDialog: AboutDialog
- backPlayer: BackSoundPlayer
- controlsDialog: ControlsDialog
- display: CanvasDisplay
- emulator: Emulator
- file: File
- fileChooser: FileChooser
- forePlayer: ForeSoundPlayer
- keyboard: KeyCode ([]) = {KeyCode.X, Key...
- mainMenuBar: MainMenuBar
- mainPane: VBox
- mainStage: Stage
- prefsDialog: PreferencesDialog
- buildFileChooser(): void
- configMenuBar(): void
- configPrefs(): void
+ main(String[]): void
- setKeyboardListener(): void
+ start(Stage): void