# Software Design Details

* 1. UML Diagrams for Utility Classes

The following diagrams represent the classes and methods within those classes that when called, execute tasks that will allow the Pacemaker to function. The diagrams also give insight into the permissions needed to access particular methods and variable values.

|  |
| --- |
| **class\_name** |
| \*\*variable: type |
| \*\*method: type |

|  |
| --- |
| **main()** |
|  |
|  |

|  |
| --- |
| **Pacemaker()** |
| leadOneInPin: enum  leadTwoInPin: enum  leadOneOutPin: enum  leadTwoOutPin: enum  maxVOut: uint16\_t  comPort: uint8\_t  txRegister: uint8\_t  rxRegister: uint8\_t |
|  |

|  |
| --- |
| **Communications() extends Pacemaker** |
| i\_CommIn: 16bytes  vraw: uint16\_t  f\_marker: uint16\_t  i\_magnet: int  i\_vs:??  o\_CommOut: uint8\_t  o\_vp:?? |
| sendEGM(): private int  initEGM(): public void |

|  |
| --- |
| **Sense() extends Pacemaker** |
| chambersSensed: private enum  activityResponse: private enum |
| setChambersSensed(enum): protected void  getChambersSensed(): public enum  setActivityResponse(enum): protected void  getActivityResponse(): public enum |

|  |
| --- |
| **Pace() extends Sense** |
| pacingState: private enum  pacingMode: private enum  hysteresis: private Boolean  hysteresisInterval: private uint16\_t  lowrateInterval: private uint16\_t  vPaceAmp: private uint16\_t  vPaceWidth: private uint16\_t  VRP: private uint16\_t |
| setPaceMode(enum): protected void  getPaceMode(): public enum  setPaceState(enum): protected void  getPaceState(): public enum  setHysteresisInterval(uint16\_t): protected void  getHysteresisInterval(): public uint16\_t  setLowRateInterval(uint16\_t): protected void  getLowRateInterval(): public uint16\_t  setvPaceAmp(uint16\_t): protected void  getvPaceAmp(): public uint16\_t  setvPaceWidth(uint16\_t): protected void  getvPaceWidth(): public uint16\_t  setVRP(uint16\_t): protected void  getVRP(): public uint16\_t |

* 1. Utility Classes

The following tables outline the public, private and protected methods making up each class defined above in section 3.1. Note that the *Sense* and *Communications* classes extend the *Pacemaker* class allowing them to inherit the properties defined in the Pacemaker class. The method *Pace* extends the *Sense* class in order to inherit properties of both Pacemaker and Sense.

Class 1: main()

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Return Type** | **Description** | **Next Action (If action event triggered)** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Class 2: Pacemaker()

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Return Type** | **Description** | **Next Action (If action event triggered)** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Class 3: Sense()

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Return Type** | **Description** | **Next Action (If action event triggered)** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |