NED University of Engineering & Technology, Karachi

Department of Software Engineering

Formal Methods in Software Engineering(SE-313)

ASSIGMMENT

Professor: Dr.Mustafa Latif

Title: Ambulatory Blood Pressure Monitor System

Group: B7

Members:

Khawar Khan(SE-21093)

Muhammad Zubair(SE-21094)

DATE: Jan 11, 2024

INDEX

I. SCOPE:

The project focuses on developing an Ambulatory Blood Pressure Monitoring (ABPM) system designed to provide continuous and accurate monitoring of blood pressure levels. The primary goal is to assist healthcare professionals in diagnosing and managing various blood pressure conditions, including Normal, Elevated, Hypertension Stage 1, Hypertension Stage 2, and Hypertensive Crisis.

Here are the general categories for blood pressure levels:

- Low Blood Pressure: Systolic < 90 mm Hg and Diastolic < 60 mm Hg
- Normal: Systolic < 120 mm Hg and Diastolic < 80 mm Hg
- Elevated: Systolic 120-129 mm Hg and Diastolic < 80 mm Hg
- Hypertension Stage 1: Systolic 130-139 mm Hg or Diastolic 80-89 mm Hg
- Hypertension Stage 2: Systolic ≥ 140 mm Hg or Diastolic ≥ 90 mm Hg
- Hypertensive Crisis: Systolic > 180 mm Hg and/or Diastolic > 120 mm Hg

The system's core functionality lies in its ability to perform automatic and regular blood pressure measurements at predetermined intervals throughout the regular period. These measurements will be categorized based on internationally recognized standards, allowing for the classification of blood pressure readings into distinct categories.

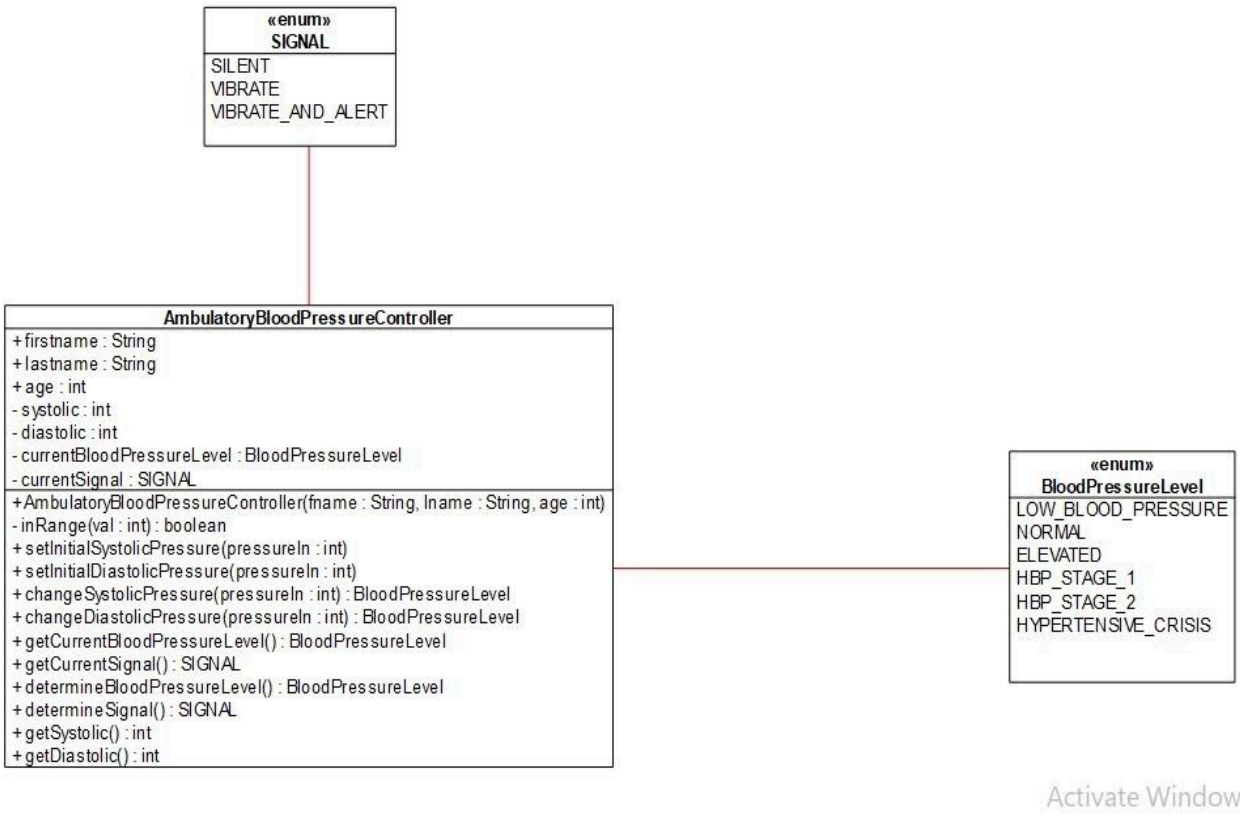
The integrity of the ABPM system is paramount, representing a cornerstone of trust for healthcare professionals in making critical decisions for patient care. The system's high level of integrity is manifested in its unwavering commitment to accurate and continuous blood pressure monitoring. Through robust algorithms and adherence to international standards, the system ensures precise categorization of blood pressure readings, contributing to an enhanced level of diagnostic precision. This reliability becomes the linchpin for healthcare practitioners, empowering them with trustworthy data that directly influences treatment plans and interventions. In this way, the system's commitment to integrity not only elevates diagnostic accuracy but also plays a central role in upholding patient safety throughout the monitoring process.

The anticipated outcomes of this ABPM system include improved diagnostic accuracy, personalized treatment plans, and continuous patient care. By providing

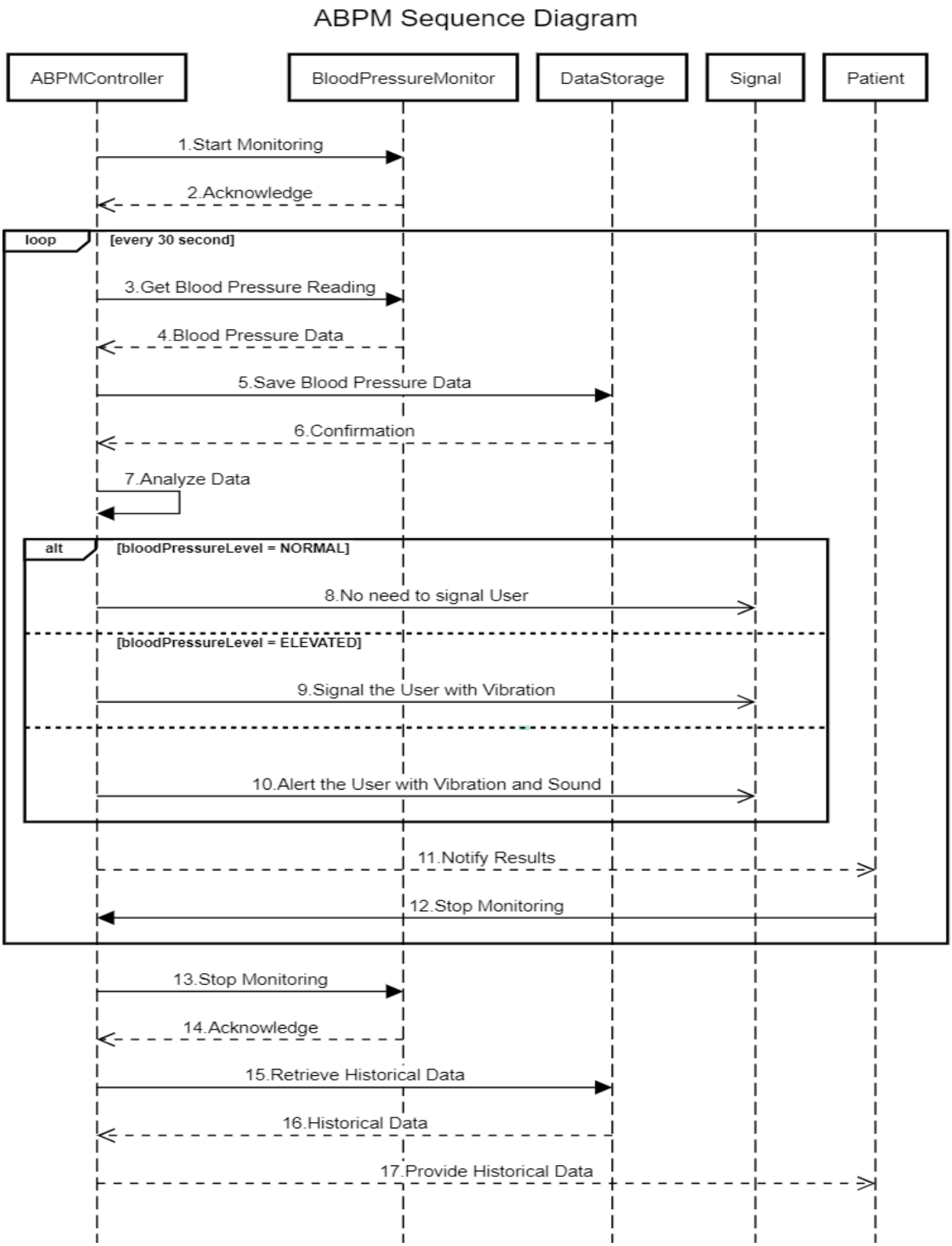
healthcare professionals with a robust tool for monitoring blood pressure, the system aims to contribute to better health outcomes and increased patient engagement in the management of their blood pressure conditions.

II. 4+1 VIEW MODEL

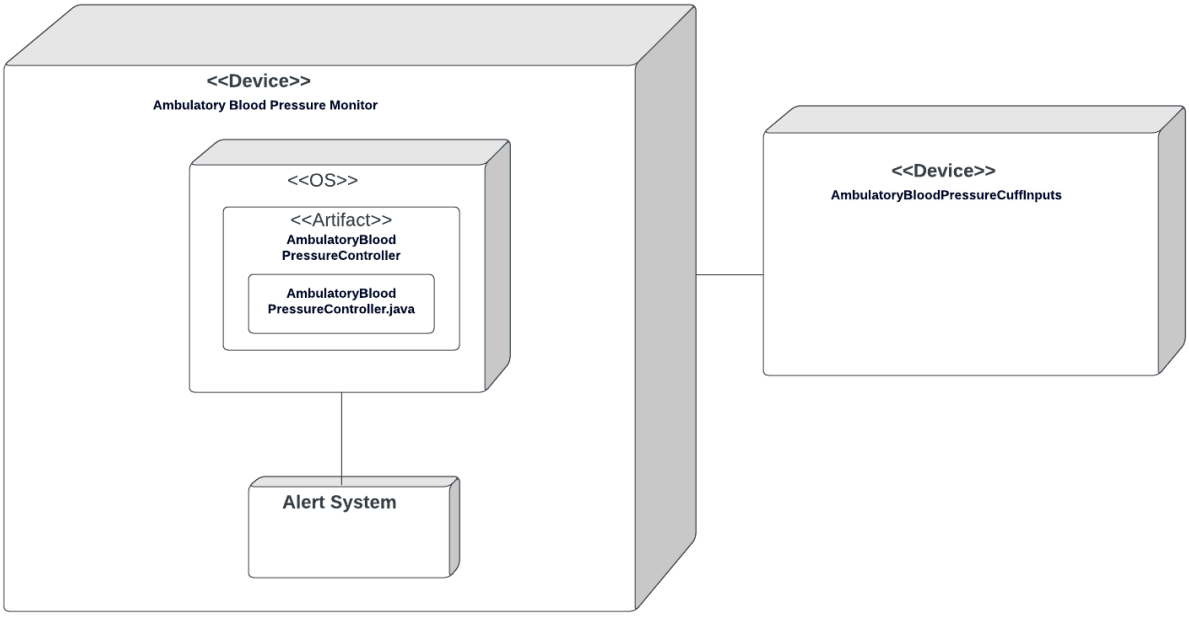
1. Logical View



2. Process View

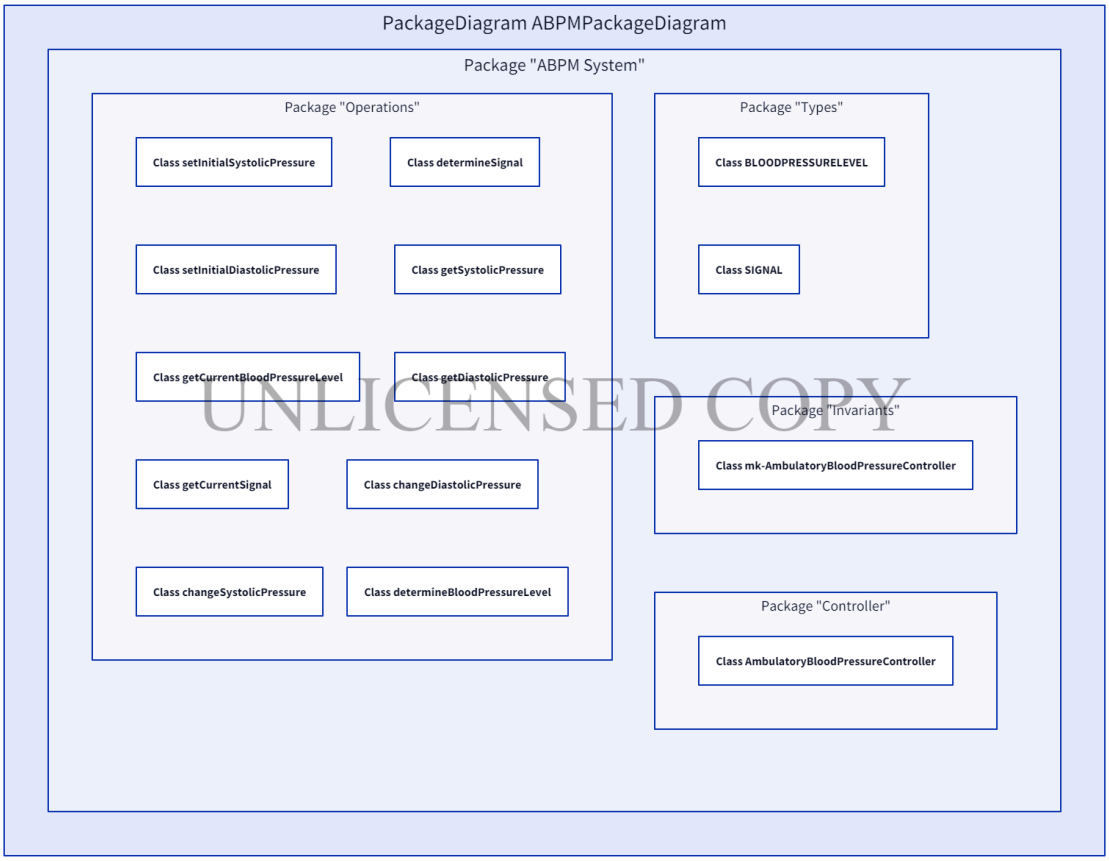


3. Physical View
Deployment Diagram

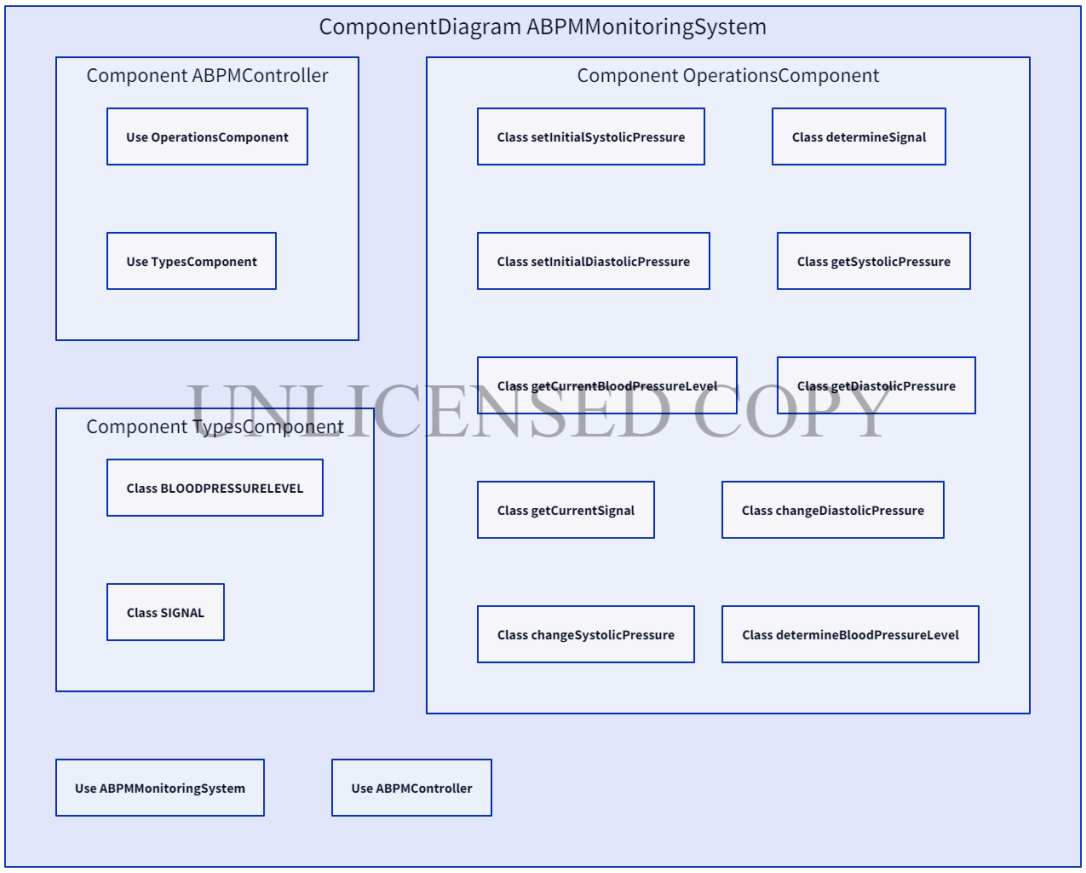


4. Development View

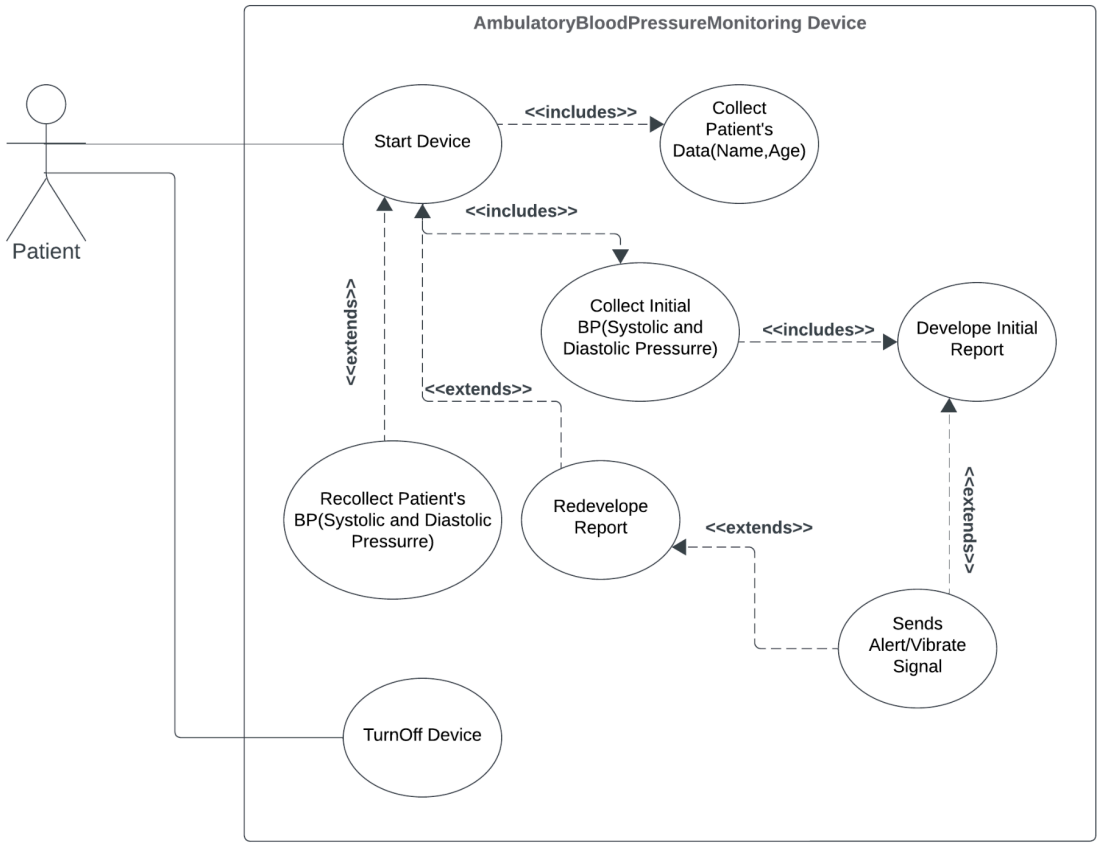
a. Package Diagram



b. Component Diagram



5. Scenarios View



III. VDM-SL SPECIFICATION OF THE SYSTEM & JAVA CODE

IV. VDM-Specification of APBM

types
SIGNAL = <SILENT> | <VIBRATE> | <VIBRATE_AND_ALERT>
BLOODPRESSURELEVEL = <LOW_BLOOD_PRESSURE> | <NORMAL> |
<ELEVATED> | <HBP_STAGE_1> | <HBP_STAGE_2> |
<HYPERTENSIVE_CRISIS>

Values
MIN_BLOOD_PRESSURE_LEVEL: Z = 1
MAX_BLOOD_PRESSURE_LEVEL: Z = 400

State *AmbulatoryBloodPressureController* **of**
systolic: [Z]
diastolic: [Z]
currentBloodPressureLevel : [BloodPressureLevel]

currentSignal : [Signal]
firstname: [seq of char]
lastname: [seq of char]
age:[Z]

inv mk-*AmbulatoryBloodPressureController* (s,d) $\triangleq$ (*MIN_BLOOD_PRESSURE_LEVEL* ≤s≤ *MAX_BLOOD_PRESSURE_LEVEL*) v s=**0**) ∧ (*MIN_BLOOD_PRESSURE_LEVEL* ≤d≤ *MAX_BLOOD_PRESSURE_LEVEL* v d=**0**)

init mk-*AmbulatoryBloodPressureController* (s,d,fn,ln,a) $\triangleq$ (s=**0**) ∧ (d=**0**) ∧ (fn=|) ∧ (an=|) ∧ (ln=|)

end

operations

inRange(val: Z) result: B
pre TRUE
post result ⇔ *MIN_BLOOD_PRESSURE_LEVEL* ≤ *val* ≤ *MAX_BLOOD_PRESSURE_LEVEL*

setInitialSystolicPressure(pressureIn:Z)
ext wr systolic: [Z]
pre inRange(pressureIn) ∧ systolic=0
post systolic = pressureIn

setInitialDiastolicPressure(pressureIn: Z)
ext wr diastolic: [Z]
pre inRange(pressureIn) ∧ diastolic=0
post diastolic = pressureIn

getCurrentBloodPressureLevel() : bloodPressureLevelOut : BloodPressureLevel
ext rd currentBloodPressureLevel
pre currentBloodPressureLevel ≠ nil
post bloodPressureLevelOut = currentBloodPressureLevel

getCurrentSignal() : signalOut : Signal
ext rd currentSignal
pre currentSignal ≠ nil
post signalOut = currentSignal

changeSystolicPressure(pressureIn) bloodPressureLevelOut: BloodPressureLevel
ext wr systolic: [Z]
pre inRange(pressureIn)
post systolic = pressureIn ∧ bloodPressureLevelOut = currentBloodPressure

changeDiastolicPressure(pressureIn) bloodPressureLevelOut: BloodPressureLevel
ext wr diastolic: [Z]
pre inRange(pressureIn)
post diastolic = pressureIn ∧ bloodPressureLevelOut = currentBloodPressure

determineBloodPressureLevel() currentBloodPressureLevelOut : BloodPressureLevel
ext rd systolic: [Z]
rd diastolic: [Z]
ext wr currentBloodPressureLevel
pre inRange(systolic) ∧ systolic ≠ 0 ∧ inRange(diastolic) ∧ diastolic≠ 0
post **((**systolic < 90 ∧ diastolic < 60 ∧ currentBloodPressureLevel = <LOW_BLOOD_PRESSURE> ∧ currentBloodPressureLevelOut = <LOW_BLOOD_PRESSURE>**)** ∨ **(** 90 ≤ systolic ≤ 120 ∧ 60≤diastolic ≤ 80 ∧ currentBloodPressureLevel = <NORMAL> ∧ currentBloodPressureLevelOut = <NORMAL>**)** ∨ **(** 120≤ systolic ≤ 130 ∧ diastolic ≤ 80 ∧ currentBloodPressureLevel = <ELEVATED> ∧ currentBloodPressureLevelOut = <ELEVATED>**)** ∨ **(** 130 < systolic ≤ 140 ∧ 80 ≤ diastolic ≤ 90 ∧ currentBloodPressureLevel = <HBP_STAGE_1> ∧ currentBloodPressureLevelOut = <HBP_STAGE_1>**)** ∨ **(** systolic > 140 ∧ diastolic > 90 ∧ currentBloodPressureLevel = <HYPERTENSIVE_CRISIS> ∧ currentBloodPressureLevelOut = <HYPERTENSIVE_CRISIS> **)** ∨ **(**currentBloodPressureLevel = <HBP_STAGE_2> ∧ currentBloodPressureLevelOut = <HBP_STAGE_2>**))**

determineSignal() signalOut : Signal
ext rd systolic: [Z]
rd diastolic: [Z]
ext wr currentSignal: [Signal]
pre inRange(systolic) ∧ systolic ≠ 0 ∧ inRange(diastolic) ∧ diastolic≠ 0
post **((**systolic < 90 ∧ diastolic < 60 ∧ currentSignal= <LOW_BLOOD_PRESSURE> ∧ signalOut = <ALERT_AND_VIBRATE>**)** ∨ **(** 90 ≤ systolic ≤ 120 ∧ 60≤diastolic ≤ 80 ∧ currentSignal= <SILENT> ∧ signalOut = <SILENT>**)** ∨ **(** 120≤ systolic ≤ 130 ∧ diastolic ≤ 80 ∧ currentSignal= <VIBRATE> ∧ signalOut = <VIBRATE>**)** ∨ **(** 130 < systolic ≤ 140 ∧ 80 ≤ diastolic ≤ 90 ∧ currentSignal= <ALERT_AND_VIBRATE> ∧ signalOut = <ALERT_AND_VIBRATE>**)** ∧ signalOut = <VIBRATE_AND_ALERT>**)** ∨ **(** systolic > 140 ∧ diastolic > 90 ∧ currentSignal= <VIBRATE_AND_ALERT> ∧ signalOut = <VIBRATE_AND_ALERT> **)** ∨ **(**currentSignal = <VIBRATE_AND_ALERT> ∧ signalOut = <VIBRATE_AND_ALERT>**))**

getSystolicPressure() pressureOut: [Z]
ext rd systolic: [Z]
pre systolic ≠ 0

post pressureOut = systolic

getDiastolicPressure() pressureOut: [Z]
ext rd diastolic: [Z]
pre diastolic ≠ 0
post pressureOut = diastolic

VDM Specification	Java Code

types
SIGNAL = <SILENT> | <VIBRATE> | <VIBRATE_AND_ALERT>
BLOODPRESSURELEVEL = <LOW_BLOOD_PRESSURE> | <NORMAL> | <ELEVATED> | <HBP_STAGE_1> | <HBP_STAGE_2> | <HYPERTENSIVE_CRISIS>
	enum BloodPressureLevel {
	LOW_BLOOD_PRESSURE,
	NORMAL,
	ELEVATED,
	HBP_STAGE_1,
	HBP_STAGE_2,
	HYPERTENSIVE_CRISIS
	}
	enum SIGNAL { SILENT, VIBRATE, VIBRATE_AND_ALERT }

—----------	----------
values *MIN_BLOOD_PRESSURE_LEVEL*: Z = 1 *MAX_BLOOD_PRESSURE_LEVEL*: Z = 400	public final int MIN_BLOOD_PRESSURE_LEVEL = 1; public static final int MAX_BLOOD_PRESSURE_LEVEL = 400;
—----------	----------
inv mk-*AmbulatoryBloodPressureController*(s,d) ≜ (*MIN_BLOOD_PRESSURE_LEVEL* ≤s≤ *MAX_BLOOD_PRESSURE_LEVEL*) v s=**0**) ∧ (*MIN_BLOOD_PRESSURE_LEVEL* ≤d≤	public boolean inv() { return ((this.systolic>=MIN_BLOOD_PRESSURE_LEVEL &&

MAX_BLOOD_PRESSURE_LEVEL v d=**0**)	this.systolic<=MAX_BLOOD_PRESSURE_LEVEL) || this.systolic==0) && ((this.diastolic>=MIN_BLOOD_PRESSURE_LEVEL && this.diastolic<=MAX_BLOOD_PRESSURE_LEVEL)|| this.systolic==0); }
—----------	—----------
init mk-*AmbulatoryBloodPressureController* (s,d,fn,ln,a) **Δ** (s=**0**) ∧ (d=**0**) ∧ (fn=|) ∧ (an=|) ∧ (ln=|)	public AmbulatoryBloodPressureController(String fname, String lname, int age) { this.firstname = fname; this.lastname = lname; this.age = age; this.systolic=0; this.diastolic=0; }
—----------	—----------
inRange(val: Z) result: B **pre** TRUE **post** result ⇔ *MIN_BLOOD_PRESSURE_LEVEL* ≤ *val* ≤ *MAX_BLOOD_PRESSURE_LEVEL*	private boolean inRange(int val) { return val >= MIN_BLOOD_PRESSURE_LEVEL && val <= MAX_BLOOD_PRESSURE_LEVEL; }
—----------	----------
setInitialSystolicPressure(pressureIn:Z) **ext wr** systolic: [Z] **pre** inRange(pressureIn) ∧ systolic=0 **post** systolic = pressureIn	public void setInitialSystolicPressure(int pressureIn) { VDM.preTest(inRange(pressureIn)&& systolic==0); this.systolic = pressureIn; VDM.invTest(this); }
—----------	----------
setInitialDiastolicPressure(pressureIn: Z) **ext wr** diastolic: [Z] **pre** inRange(pressureIn) ∧ diastolic=0 **post** diastolic = pressureIn	public void setInitialDiastolicPressure(int pressureIn) { VDM.preTest(inRange(pressureIn)&& diastolic==0); this.diastolic = pressureIn; VDM.invTest(this); }
—----------	----------
determineBloodPressureLevel() currentBloodPressureLevelOut :	public BloodPressureLevel

```
BloodPressureLevel
ext rd systolic: [Z]
     rd diastolic: [Z]
ext wr currentBloodPressureLevel
pre     inRange(systolic) ∧ systolic ≠ 0  ∧
inRange(diastolic) ∧ diastolic≠ 0
post    ((systolic < 90 ∧ diastolic < 60 ∧
currentBloodPressureLevel =
        <LOW_BLOOD_PRESSURE>  ∧
        currentBloodPressureLevelOut =
        <LOW_BLOOD_PRESSURE>) v (  90 ≤
        systolic ≤  120 ∧  60≤diastolic ≤ 80 ∧
        currentBloodPressureLevel =
        <NORMAL> ∧
        currentBloodPressureLevelOut =
        <NORMAL>) v (  120≤ systolic ≤  130 ∧
        diastolic ≤ 80 ∧
        currentBloodPressureLevel =
        <ELEVATED>  ∧
        currentBloodPressureLevelOut =
        <ELEVATED>)  v (  130 < systolic ≤  140
        ∧ 80 ≤ diastolic ≤ 90 ∧
        currentBloodPressureLevel =
        <HBP_STAGE_1>  ∧
        currentBloodPressureLevelOut =
        <HBP_STAGE_1>)  v (  systolic >  140
        ∧ diastolic > 90 ∧
        currentBloodPressureLevel =
        <HYPERTENSIVE_CRISIS>  ∧
        currentBloodPressureLevelOut =
        <HYPERTENSIVE_CRISIS> )  v
        (currentBloodPressureLevel =
        <HBP_STAGE_2>  ∧
        currentBloodPressureLevelOut =
        <HBP_STAGE_2>))

—----------------------------------------------------

determineSignal() signalOut : Signal
ext rd systolic: [Z]
     rd diastolic: [Z]
ext wr currentSignal: [Signal]
pre     inRange(systolic) ∧ systolic ≠ 0  ∧
inRange(diastolic) ∧ diastolic≠ 0
post    ((systolic < 90 ∧ diastolic < 60 ∧
currentSignal=
```

```
determineBloodPressureLevel() {

  VDM.preTest(inRange(systolic) &&
inRange(diastolic) && systolic!=0 && diastolic!=0);

  if (systolic < 90 && diastolic < 60) {
    currentBloodPressureLevel =
BloodPressureLevel.LOW_BLOOD_PRESSURE;
  } else if (systolic >= 90 && systolic <= 120 &&
diastolic >= 60 && diastolic <= 80) {
    currentBloodPressureLevel =
BloodPressureLevel.NORMAL;
  } else if (systolic > 120 && systolic <= 130 &&
diastolic <= 80) {
    currentBloodPressureLevel =
BloodPressureLevel.ELEVATED;
  } else if (systolic > 130 && systolic <= 140 &&
diastolic > 80 && diastolic <= 90) {
    currentBloodPressureLevel =
BloodPressureLevel.HBP_STAGE_1;
  } else if (systolic > 140 && diastolic > 90) {
    currentBloodPressureLevel =
BloodPressureLevel.HYPERTENSIVE_CRISIS;
  } else {
    currentBloodPressureLevel =
BloodPressureLevel.HBP_STAGE_2;
  }

  VDM.invTest(this);

  return currentBloodPressureLevel;
}
```

```
public SIGNAL determineSignal() {
  boolean validInputRange = inRange(systolic) &&
inRange(diastolic) && systolic != 0 && diastolic !=
0;
```

<LOW_BLOOD_PRESSURE> ∧ signalOut = <ALERT_AND_VIBRATE>) v (90 ≤ systolic ≤ 120 ∧ 60≤diastolic ≤ 80 ∧ currentSignal= <SILENT> ∧ signalOut = <SILENT>) v (120≤ systolic ≤ 130 ∧ diastolic ≤ 80 ∧ currentSignal= <VIBRATE> ∧ signalOut = <VIBRATE>) v (130 < systolic ≤ 140 ∧ 80 ≤ diastolic ≤ 90 ∧ currentSignal= <ALERT_AND_VIBRATE> ∧ signalOut = <ALERT_AND_VIBRATE>) ∧ signalOut = <VIBRATE_AND_ALERT>) v (systolic > 140 ∧ diastolic > 90 ∧ currentSignal= <VIBRATE_AND_ALERT> ∧ signalOut = <VIBRATE_AND_ALERT>) v (currentSignal = <VIBRATE_AND_ALERT> ∧ signalOut = <VIBRATE_AND_ALERT>))	VDM.preTest(validInputRange);
	
	if (systolic < 90 && diastolic < 60) {
	currentSignal = SIGNAL.VIBRATE_AND_ALERT;
	} else if (systolic >= 90 && systolic <= 120 && diastolic >= 60 && diastolic <= 80) {
	currentSignal = SIGNAL.SILENT;
	} else if (systolic > 120 && systolic <= 130 && diastolic <= 80) {
	currentSignal = SIGNAL.VIBRATE;
	} else if (systolic > 130 && systolic <= 140 && diastolic > 80 && diastolic <= 90) {
	currentSignal = SIGNAL.VIBRATE_AND_ALERT;
	} else if (systolic > 140 && diastolic > 90) {
	currentSignal = SIGNAL.VIBRATE_AND_ALERT;
	} else {
	currentSignal = SIGNAL.VIBRATE_AND_ALERT;
	}
	
	VDM.invTest(this);
	
	return currentSignal;
	}

V. JAVA CODE OF ABPM system

```
package ABPM;

import FMS.VDM;

enum BloodPressureLevel {

 LOW_BLOOD_PRESSURE,

 NORMAL,
```

```
  ELEVATED,
  HBP_STAGE_1,
  HBP_STAGE_2,
  HYPERTENSIVE_CRISIS
}
enum SIGNAL { SILENT, VIBRATE, VIBRATE_AND_ALERT }
interface invChecker {
  public boolean inv();
}
public class AmbulatoryBloodPressureController implements invChecker{
  public String firstname, lastname;
  public int age;
  private int systolic, diastolic;
  private BloodPressureLevel currentBloodPressureLevel = null;
  private SIGNAL currentSignal = null;

  public final int MIN_BLOOD_PRESSURE_LEVEL = 1;
  public static final int MAX_BLOOD_PRESSURE_LEVEL = 400;

  public AmbulatoryBloodPressureController(
    String fname, String lname, int age) {
    this.firstname = fname;
    this.lastname = lname;
    this.age = age;
    this.systolic=0;
    this.diastolic=0;
```

```
 }

 @Override

 public boolean inv() {

                                    return ((this.systolic>=MIN_BLOOD_PRESSURE_LEVEL &&
this.systolic<=MAX_BLOOD_PRESSURE_LEVEL)|| this.systolic==0) &&
((this.diastolic>=MIN_BLOOD_PRESSURE_LEVEL && this.diastolic<=MAX_BLOOD_PRESSURE_LEVEL)||
this.systolic==0);

 }

 private boolean inRange(int val) {

  return val >= MIN_BLOOD_PRESSURE_LEVEL && val <= MAX_BLOOD_PRESSURE_LEVEL;

 }

 public void setInitialSystolicPressure(int pressureIn) {

  try {

   VDM.preTest(inRange(pressureIn)&& systolic==0);

   this.systolic = pressureIn;

  } catch (VDM.VDMPreconditionException e) {

   System.err.println("Precondition test failed: "

      + "Initial systolic Value given is out of Range");

      this.systolic = 0;

  }

  try{

   VDM.invTest(this);

  }catch(Exception e) {

   System.err.println("Postcondition test failed: "

      + e.getMessage());

  }

 }
```

```
public void setInitialDiastolicPressure(int pressureIn) {
  try {
    VDM.preTest(inRange(pressureIn));
    this.diastolic = pressureIn;
  } catch (VDM.VDMPreconditionException e) {
    System.err.println("Precondition test failed: "
        + "Initial diastolic Value given is out of Range");
     this.diastolic = 0;
  }
  try{
    VDM.invTest(this);
  }catch(Exception e) {
    System.err.println("Postcondition test failed: "
        + e.getMessage());
  }
}
public BloodPressureLevel changeSystolicPressure(int pressureIn) {
  try {
    VDM.preTest(inRange(pressureIn));
    this.systolic = pressureIn;
  } catch (VDM.VDMPreconditionException e) {
    System.err.println("Precondition test failed: "
        + "Given systolic Value given is out of Range");
    this.systolic = 0;
  }
  try{
```

```
    VDM.invTest(this);
   }catch(Exception e) {
    System.err.println("Postcondition test failed: "
        + e.getMessage());
   }
   return determineBloodPressureLevel();
  }
  public BloodPressureLevel changeDiastolicPressure(int pressureIn) {
   try {
    VDM.preTest(inRange(pressureIn));
    this.diastolic = pressureIn;
   } catch (VDM.VDMPreconditionException e) {
    System.err.println("Precondition test failed: "
        + "Given diastolic Value given is out of Range");
    this.diastolic=0;
   }
  try{
    VDM.invTest(this);
   }catch(Exception e) {
    System.err.println("Postcondition test failed: "
        + e.getMessage());
   }
   return determineBloodPressureLevel();
  }
  public BloodPressureLevel getCurrentBloodPressureLevel() {
   try {
```

```
    VDM.preTest(currentBloodPressureLevel != null);
    return currentBloodPressureLevel;
   } catch (VDM.VDMPreconditionException e) {
    System.err.println("Precondition test failed: "
        + "CurrentBloodPressue Level not Intialized");
   }
   try{
    VDM.invTest(this);
   }catch(Exception e) {
    System.err.println("Postcondition test failed: "
        + e.getMessage());
   }
   return null;
  }
  public SIGNAL getCurrentSignal() {
   try {
    VDM.preTest(currentSignal != null);
    return currentSignal;
   } catch (VDM.VDMPreconditionException e) {
    System.err.println("Precondition test failed: "
        + "CurrentBloodPressure Level not Intialized");
   }
   try{
    VDM.invTest(this);
   }catch(Exception e) {
    System.err.println("Postcondition test failed: "
```

```
        + e.getMessage());
    }
    return null;
  }
  public BloodPressureLevel determineBloodPressureLevel() {
    if (systolic < 90 && diastolic < 60) {
      try {
        VDM.preTest(inRange(systolic) && inRange(diastolic) && systolic!=0 && diastolic!=0);
        currentBloodPressureLevel = BloodPressureLevel.LOW_BLOOD_PRESSURE;
      } catch (VDM.VDMPreconditionException e) {
        System.err.println("Precondition test failed: " + e.getMessage());
      }
      try{
      VDM.invTest(this);
    }catch(Exception e) {
      System.err.println("Postcondition test failed: "
          + e.getMessage());
    }
      return currentBloodPressureLevel;
    } else if (systolic >= 90 && systolic <= 120 && diastolic >= 60
        && diastolic <= 80) {
      try {
        VDM.preTest(inRange(systolic) && inRange(diastolic) && systolic!=0 && diastolic!=0);
        currentBloodPressureLevel = BloodPressureLevel.NORMAL;
      } catch (VDM.VDMPreconditionException e) {
        System.err.println("Precondition test failed: " + e.getMessage());
```

```
    }
     try{
    VDM.invTest(this);
   }catch(Exception e) {
    System.err.println("Postcondition test failed: "
        + e.getMessage());
   }
    return currentBloodPressureLevel;
   } else if (systolic > 120 && systolic <= 130 && diastolic <= 80) {
    try {
      VDM.preTest(inRange(systolic) && inRange(diastolic) && systolic!=0 && diastolic!=0);
      currentBloodPressureLevel = BloodPressureLevel.ELEVATED;
    } catch (VDM.VDMPreconditionException e) {
      System.err.println("Precondition test failed: " + e.getMessage());
    }
     try{
    VDM.invTest(this);
   }catch(Exception e) {
    System.err.println("Postcondition test failed: "
        + e.getMessage());
   }
    return currentBloodPressureLevel;
   } else if (systolic > 130 && systolic <= 140 && diastolic > 80
      && diastolic <= 90) {
    try {
      VDM.preTest(inRange(systolic) && inRange(diastolic) && systolic!=0 && diastolic!=0);
```

```
      currentBloodPressureLevel = BloodPressureLevel.HBP_STAGE_1;
    } catch (VDM.VDMPreconditionException e) {
      System.err.println("Precondition test failed: " + e.getMessage());
    }
    try{
    VDM.invTest(this);
  }catch(Exception e) {
    System.err.println("Postcondition test failed: "
        + e.getMessage());
  }
    return currentBloodPressureLevel;
  } else if (systolic > 140 && diastolic > 90) {
    try {
      VDM.preTest(inRange(systolic) && inRange(diastolic) && systolic!=0 && diastolic!=0);
      currentBloodPressureLevel = BloodPressureLevel.HYPERTENSIVE_CRISIS;
    } catch (VDM.VDMPreconditionException e) {
      System.err.println("Precondition test failed: " + e.getMessage());
    }
    try{
    VDM.invTest(this);
  }catch(Exception e) {
    System.err.println("Postcondition test failed: "
        + e.getMessage());
  }
    return currentBloodPressureLevel;
  } else {
```

```
    try {
      VDM.preTest(inRange(systolic) && inRange(diastolic) && systolic!=0 && diastolic!=0);
      currentBloodPressureLevel = BloodPressureLevel.HBP_STAGE_2;
    } catch (VDM.VDMPreconditionException e) {
      System.err.println("Precondition test failed: " + e.getMessage());
    }
    try{
    VDM.invTest(this);
   }catch(Exception e) {
    System.err.println("Postcondition test failed: "
        + e.getMessage());
   }
    return currentBloodPressureLevel;
   }
 }
 public SIGNAL determineSignal() {
  if (systolic < 90 && diastolic < 60) {
    try {
      VDM.preTest(inRange(systolic) && inRange(diastolic) && systolic!=0 && diastolic!=0);
      currentSignal = SIGNAL.VIBRATE_AND_ALERT;
    } catch (VDM.VDMPreconditionException e) {
      System.err.println("Precondition test failed: " + e.getMessage());
    }
    try{
    VDM.invTest(this);
   }catch(Exception e) {
```

```
    System.err.println("Postcondition test failed: "
        + e.getMessage());
  }
    return currentSignal;
  } else if (systolic >= 90 && systolic <= 120 && diastolic >= 60
     && diastolic <= 80) {
    try {
     VDM.preTest(inRange(systolic) && inRange(diastolic) && systolic!=0 && diastolic!=0);
     currentSignal = SIGNAL.SILENT;
    } catch (VDM.VDMPreconditionException e) {
     System.err.println("Precondition test failed: " + e.getMessage());
    }
    try{
    VDM.invTest(this);
  }catch(Exception e) {
    System.err.println("Postcondition test failed: "
        + e.getMessage());
  }
    return currentSignal;
  } else if (systolic > 120 && systolic <= 130 && diastolic <= 80) {
    try {
     VDM.preTest(inRange(systolic) && inRange(diastolic) && systolic!=0 && diastolic!=0);
     currentSignal = SIGNAL.VIBRATE;
    } catch (VDM.VDMPreconditionException e) {
     System.err.println("Precondition test failed: " + e.getMessage());
    }
```

```
    try{
    VDM.invTest(this);
  }catch(Exception e) {
    System.err.println("Postcondition test failed: "
        + e.getMessage());
  }
    return currentSignal;
  } else if (systolic > 130 && systolic <= 140 && diastolic > 80
     && diastolic <= 90) {
    try {
      VDM.preTest(inRange(systolic) && inRange(diastolic) && systolic!=0 && diastolic!=0);
      currentSignal = SIGNAL.VIBRATE_AND_ALERT;
    } catch (VDM.VDMPreconditionException e) {
      System.err.println("Precondition test failed: " + e.getMessage());
    }
    try{
    VDM.invTest(this);
  }catch(Exception e) {
    System.err.println("Postcondition test failed: "
        + e.getMessage());
  }
    return currentSignal;
  } else if (systolic > 140 && diastolic > 90) {
    try {
      VDM.preTest(inRange(systolic) && inRange(diastolic) && systolic!=0 && diastolic!=0);
      currentSignal = SIGNAL.VIBRATE_AND_ALERT;
```

```
    } catch (VDM.VDMPreconditionException e) {
      System.err.println("Precondition test failed: " + e.getMessage());
    }
    try{
    VDM.invTest(this);
   }catch(Exception e) {
    System.err.println("Postcondition test failed: "
        + e.getMessage());
   }
    return currentSignal;
   } else {
    try {
      VDM.preTest(inRange(systolic) && inRange(diastolic) && systolic!=0 && diastolic!=0);
      currentSignal = SIGNAL.VIBRATE_AND_ALERT;
    } catch (VDM.VDMPreconditionException e) {
      System.err.println("Precondition test failed: " + e.getMessage());
    }
    try{
    VDM.invTest(this);
   }catch(Exception e) {
    System.err.println("Postcondition test failed: "
        + e.getMessage());
   }
    return currentSignal;
   }
  }
```

```
public int getSystolic() {
  try {
    VDM.preTest(systolic!=0);
  } catch (VDM.VDMPreconditionException e) {
    System.err.println("Precondition test failed: " + "Stored Systolic Measure is out of Bound Or not initialized");
  }
  try{
    VDM.invTest(this);
  }catch(Exception e) {
    System.err.println("Postcondition test failed: "
        + e.getMessage());
  }
  return systolic;
}
public int getDiastolic() {
  try {
    VDM.preTest(diastolic!=0);
  } catch (VDM.VDMPreconditionException e) {
    System.err.println("Precondition test failed: " + "Stored Diastolic Measure is out of Bound Or not initialized");
  }
  try{
    VDM.invTest(this);
  }catch(Exception e) {
    System.err.println("Postcondition test failed: "
        + e.getMessage());
  }
```

```
    return diastolic;
}
}
```

VI. TEST CASES SUMMARY

Test Case ID	Test Case	Input	Expected Output	Actual Output
01	InitializingSystolicPressure_InvalidInput	pressureIn = 500	Exception Thrown: Pre-condition failed. And systolic=0	Exception: Pre-condition failed. systolic=0
02	InitializingSystolicPressure_ValidInput	pressureIn=150	systolic=150	systolic=150
03	InitializingDiastolicPressure_InvalidInput	pressureIn=450	Exception Thrown: Pre-condition failed. And diastolic=0	Exception: Pre-condition failed. diastolic=0
04	InitializingDiastolicPressure_ValidInput	pressureIn=100	diastolic=100	diastolic=100
05	ChangingSystolicPressure_InvalidInput	pressureIn=410	Exception Thrown: Pre-condition Failed. And Systolic does not change.	Exception: Pre-condition failed.
06	ChangingSystolicPressure_ValidInput	pressureIn=200	systolic=200	systolic=200
07	ChangingDiastolic	pressureIn=460	Exception	Exception:

	Pressure_InvalidInput		Thrown: Pre-condition Failed. And Diastolic does not change.	Pre-condition failed.
08	ChangingDiastolicPressure_ValidInput	pressureIn=160	diastolic=160	diastolic=160
09	GetBloodPressureLevel_AfterInitialization	Systolic and diastolic are initialized. And Systolic=100 Diastolic=70	BloodPressure=<NORMAL>	BloodPressure=<NORMAL>
10	GetBloodPressureLevel_BeforeInitialization	Systolic and diastolic are not initialized.	Exception Thrown: Pre-condition Failed.	Exception: Pre-condition failed.
11	GetSystolicPressure_AfterInitilization	Systolic is initialized. And systolic=70	pressureOut=70	pressureOut=70
12	GetSystolicPressure_BeforeInitialization	Systolic is not initialized.	Exception Thrown: Pre-condition Failed.	Exception: Pre-condition failed.
13	GetDiastolicPressure_AfterInitilization	Diastolic is initialized. And diastolic=70	pressureOut=70	pressureOut=70
14	GetDiastolicPressure_BeforeInitialization	Diastolic is not initialized.	Exception Thrown: Pre-condition Failed.	Exception: Pre-condition failed.
15	DetermineBloodPressure_LOWBP_AND_DetermineSig	systolic=85 diastolic=55	bloodPressureOut = <LOW_BLOOD	bloodPressureOut = LOW_BLOOD_

	nal_<ALERT_AND_VIBRATE>		_PRESSURE>	PRESSURE
16	DetermineBloodPressure_NORMAL_AND_DetermineSignal_<SILENT>	systolic=100 diastolic=70	bloodPressureOut = <NORMAL> AND signalOut=<SILENT>	bloodPressureOut = NORMAL signalOut=<SILENT>
17	DetermineBloodPressure_ELEVATED_AND_DetermineSignal_<VIBRATE>	systolic=121 diastolic=78	bloodPressureOut = <ELEVATED> AND signalOut=<VIBRATE>	bloodPressureOut = ELEVATED signalOut=<VIBRATE>
18	DetermineBloodPressure_HBPSTAGE1_AND_DetermineSignal_<ALERT_AND_VIBRATE>	systolic=135 diastolic=85	bloodPressureOut = <HBP_STAGE1> AND signalOut=<ALERT_AND_VIBRATE>	bloodPressureOut = HBP_STAGE1 signalOut=<ALERT_AND_VIBRATE>
19	DetermineBloodPressure_HBPSTAGE2_AND_DetermineSignal_<ALERT_AND_VIBRATE>	systolic=145 diastolic=85	bloodPressureOut = <HBP_STAGE2> AND signalOut=<ALERT_AND_VIBRATE>	bloodPressureOut = HBP_STAGE2 signalOut=<ALERT_AND_VIBRATE>
20	DetermineBloodPressure_HYPERTENSIVECRISIS_AND_DetermineSignal_<ALERT_AND_VIBRATE>	systolic=145 diastolic=100	bloodPressureOut = <HYPERTENSIVE_CRISIS> And signalOut=<ALERT_AND_VIBRATE>	bloodPressureOut = HYPERTENSIVE_CRISIS signalOut=<ALERT_AND_VIBRATE>

VII. TESTER CLASS JAVA CODE :

```
import java.util.Scanner;

import ABPM.AmbulatoryBloodPressureController;

public class Tester {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      AmbulatoryBloodPressureController abpc =
            new AmbulatoryBloodPressureController("John", "Doe", 62);

      int choice;

      do {
         System.out.println("\n----- Tester Menu -----");
         System.out.println("1. InitializingSystolicPressure_InvalidInput");
         System.out.println("2. InitializingSystolicPressure_ValidInput");
         System.out.println("3. InitializingDiastolicPressure_InvalidInput");
         System.out.println("4. InitializingDiastolicPressure_ValidInput");
         System.out.println("5. ChangingSystolicPressure_InvalidInput");
         System.out.println("6. ChangingSystolicPressure_ValidInput");
         System.out.println("7. ChangingDiastolicPressure_InvalidInput");
         System.out.println("8. ChangingDiastolicPressure_ValidInput");
         System.out.println("9. GetBloodPressureLevel_BeforeInitialization");
         System.out.println("10. GetSystolicPressure_AfterInitilization");
         System.out.println("11. GetSystolicPressure_BeforeInitialization");
         System.out.println("12. GetDiastolicPressure_AfterInitilization");
         System.out.println("13. GetDiastolicPressure_BeforeInitialization");
         System.out.println("14. DetermineBloodPressure_LOWBP &&
DetermineSignal_<ALERT_AND_VIBRATE>");
         System.out.println("15. DetermineBloodPressure_NORMAL &&
DetermineSignal_<SILENT>");
         System.out.println("16. DetermineBloodPressure_ELEVATED &&
DetermineSignal_<VIBRATE>");
         System.out.println("17. DetermineBloodPressure_HBPSTAGE1 &&
DetermineSignal_<ALERT_AND_VIBRATE>");
         System.out.println("18. DetermineBloodPressure_HBPSTAGE2 &&
DetermineSignal_<ALERT_AND_VIBRATE>");
```

```
        System.out.println("19. DetermineBloodPressure_HYPERTENSIVECRISIS &&
DetermineSignal_<ALERT_AND_VIBRATE>");
        System.out.println("0. Exit");

        System.out.print("Enter your choice: ");
        choice = scanner.nextInt();

        switch (choice) {
           case 1:
              System.out.println("Expected Output: Exception thrown and Systolic set to 0");
              abpc.setInitialSystolicPressure(500);
              System.out.println("Actual Output: "+abpc.getSystolic());
              break;

           case 2:
              System.out.println("Expected Output: Systolic set to 150");
              abpc.setInitialSystolicPressure(150);
              System.out.println("Actual Output: "+abpc.getSystolic());
              break;

           case 3:
              System.out.println("Expected Output: Exception thrown and Diastolic set to 0");
              abpc.setInitialDiastolicPressure(450);
              System.out.println("Actual Output: "+abpc.getDiastolic());
              break;

           case 4:
              System.out.println("Expected Output: Diastolic set to 100");
              abpc.setInitialDiastolicPressure(100);
              System.out.println("Actual Output: "+abpc.getDiastolic());
              break;

           case 5:
              System.out.println("Expected Output: Exception Thrown: Pre-condition Failed." +
                    "Systolic does not Changed (0)");
              abpc.changeSystolicPressure(410);
              System.out.println("Actual Output: "+abpc.getSystolic());
              break;

           case 6:
```

```
            System.out.println("Expected Output: Systolic = 200");
            abpc.changeSystolicPressure(200);
            System.out.println("Actual Output: "+abpc.getSystolic());
            break;

        case 7:
            System.out.println("Expected Output: Exception Thrown: Pre-condition Failed." +
                  "Diastolic does not Changed (0)");
            abpc.changeDiastolicPressure(460);
            System.out.println("Actual Output: "+abpc.getDiastolic());
            break;

        case 8:
            System.out.println("Expected Output: Systolic = 160");
            abpc.changeDiastolicPressure(160);
            System.out.println("Actual Output: "+abpc.getDiastolic());
            break;

        case 9:
                                        AmbulatoryBloodPressureController newone =new
AmbulatoryBloodPressureController("John", "Doe", 62);
            System.out.println("Expected Output: Exception Thrown: Pre-condition Failed.");
            System.out.println("Actual Output: ");
            newone.getCurrentBloodPressureLevel();
            break;

        case 10:
                    System.out.println("Expected Output: Systolic is initialized." + "And" +
"systolic=70" + "");
            abpc.setInitialSystolicPressure(70);
            System.out.println("Actual Output: "+abpc.getSystolic());
            break;

        case 11:
                                        AmbulatoryBloodPressureController newtwo =new
AmbulatoryBloodPressureController("John", "Doe", 62);
            System.out.println("Exception Thrown: Pre-condition Failed.");
            System.out.println("Actual Output: "+newtwo.getSystolic());
            break;
```

```
            case 12:
                System.out.println("Expected Output: Diastolic  is initialized." + "And" + "Diastolic
=70" + "");
                abpc.setInitialDiastolicPressure(70);
                System.out.println("Actual Output: "+abpc.getDiastolic());
                break;

            case 13:
                                                AmbulatoryBloodPressureController newthree =new
AmbulatoryBloodPressureController("John", "Doe", 62);
                System.out.println("Expected Output: Exception Thrown: Pre-condition Failed.");
                System.out.println("Actual Output: "+newthree.getDiastolic());
                break;

            case 14:
                                        System.out.println("Expected Output: bloodPressureOut =
<LOW_BLOOD_PRESSURE> and Signal: <ALERT_AND_VIBRATE>");
                abpc.setInitialSystolicPressure(150);
                abpc.setInitialDiastolicPressure(100);
                abpc.changeDiastolicPressure(55);
                abpc.changeSystolicPressure(85);
                        System.out.println("Actual Output: "+abpc.determineBloodPressureLevel()+"
"+abpc.determineSignal());
                break;

            case 15:
                System.out.println("Expected Output: bloodPressureOut = <NORMAL> and Signal:
<SILENT>");
                abpc.setInitialSystolicPressure(150);
                abpc.setInitialDiastolicPressure(100);
                abpc.changeDiastolicPressure(70);
                abpc.changeSystolicPressure(100);
                        System.out.println("Actual Output: "+abpc.determineBloodPressureLevel()+"
"+abpc.determineSignal());
                break;

            case 16:
                       System.out.println("Expected Output: bloodPressureOut = <ELEVATED> and
Signal: <VIBRATE>");
                abpc.setInitialSystolicPressure(150);
```

```
            abpc.setInitialDiastolicPressure(100);
            abpc.changeDiastolicPressure(78);
            abpc.changeSystolicPressure(121);
                System.out.println("Actual Output: "+abpc.determineBloodPressureLevel()+"
"+abpc.determineSignal());
            break;

        case 17:
            System.out.println("Expected Output: bloodPressureOut = <HBP_STAGE1> and
Signal: <ALERT_AND_VIBRATE>");
            abpc.setInitialSystolicPressure(150);
            abpc.setInitialDiastolicPressure(100);
            abpc.changeDiastolicPressure(85);
            abpc.changeSystolicPressure(135);
                System.out.println("Actual Output: "+abpc.determineBloodPressureLevel()+"
"+abpc.determineSignal());
            break;

        case 18:
            System.out.println("Expected Output: bloodPressureOut = <HBP_STAGE2> and
Signal: <ALERT_AND_VIBRATE>");
            abpc.setInitialSystolicPressure(150);
            abpc.setInitialDiastolicPressure(100);
            abpc.changeDiastolicPressure(85);
            abpc.changeSystolicPressure(145);
                System.out.println("Actual Output: "+abpc.determineBloodPressureLevel()+"
"+abpc.determineSignal());
            break;

        case 19:
                                System.out.println("Expected Output: bloodPressureOut =
<HYPERTENSIVECRISIS> and Signal: <ALERT_AND_VIBRATE>");
            abpc.setInitialSystolicPressure(150);
            abpc.setInitialDiastolicPressure(100);
            abpc.changeDiastolicPressure(100);
            abpc.changeSystolicPressure(145);
                System.out.println("Actual Output: "+abpc.determineBloodPressureLevel()+"
"+abpc.determineSignal());
            break;
```

```
            case 0:
                System.out.println("Shutting Down the device. Goodbye!");
                break;

            default:
                System.out.println("Invalid choice. Please try again.");
                break;
        }

    } while (choice != 0);
  }
}
```