



# Insulin Pump Simulation

Alexsandria Ryan  
NSCC Capstone - Fall 2023  
December 08, 2023

# Initial Project Idea

## Phase 1



- Create a simulated insulin pump in Java
- Replicate all the math and functionality that today's average pump can do



## Phase 2



- Create a mobile application that is linked to the insulin pump / insulin pump simulation
- Read blood glucose from Continuous Glucose Monitors (via NFC)
- Insulin administration from the app to the pump via Bluetooth





# Presentation Topics



What is an insulin pump?



Why try to replicate an insulin pump?



Code

Changes & Conclusions



Resources

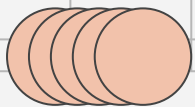


01

# What is an insulin pump?

And why they're  
important for diabetics!





# What Does an Insulin Pump Do?



Insulin pumps help to administer....

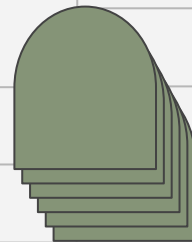
## Bolus Insulin

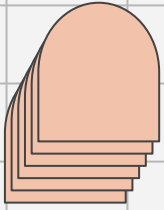
counteracts any net carbohydrates that the user consumes.



## Basal Insulin

counteracts any glucose created by the liver.

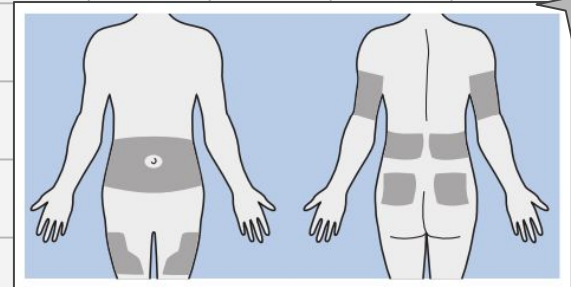
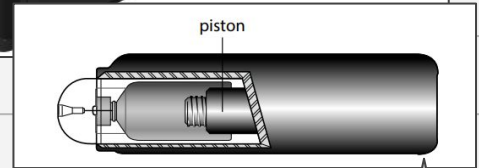
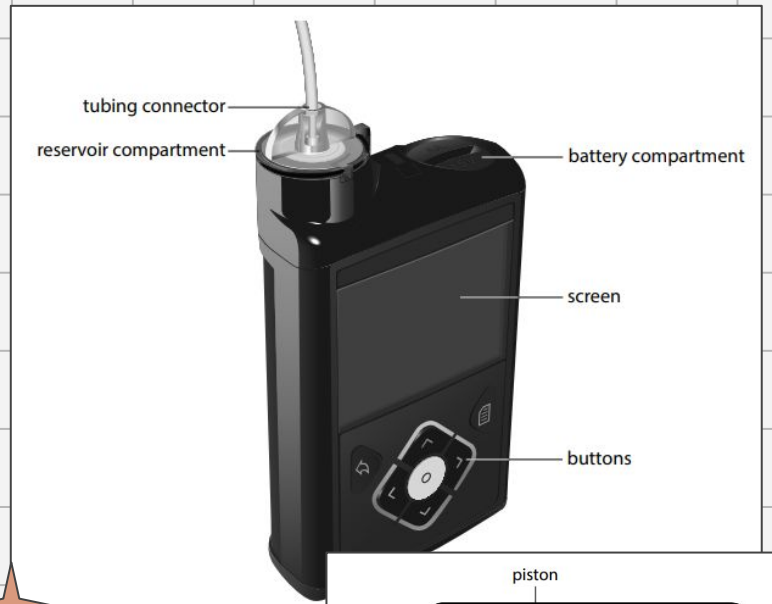


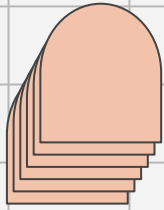


**Note:** insulin pumps come in many shapes & forms. This example is of a 'tethered' pump, a Medtronic 630G

# Anatomy of an Insulin Pump

- **Reservoir:** holds up to 3mL (300 Units) of insulin
- **Tubing Connector:** where the reservoir meets the tubing; the insulin is delivered by the tube to a **site** (which can be applied anywhere in the photo below)
- **Piston:** pushes the insulin up from the reservoir into the tubing





# Insulin Pump UI

- Examples of what the UI looks like on the Medtronic 630G
- Startup (date & time)
- Basal insulin pattern (called 'Workday')
- Bolus Wizard
  - Note: the blood glucose reading is in US format
- Main Menu



Startup 2/3

Enter Time

Time 12:00 AM

Next



Startup 3/3

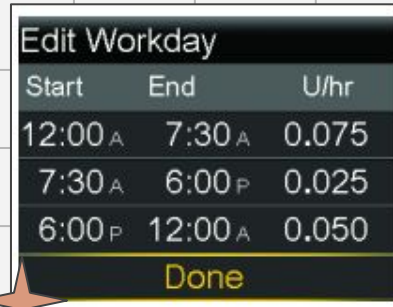
Enter Date

Year 2017

Month Jan

Day 16, Mon

Next



Edit Workday

Start	End	U/hr
12:00 A	7:30 A	0.075
7:30 A	6:00 P	0.025
6:00 P	12:00 A	0.050

Done



Bolus Wizard

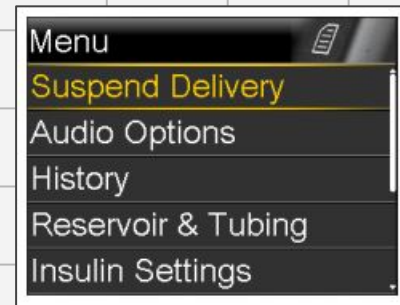
BG 130 mg/dL 9:08 AM

Active Ins. adjust. -0.3 U

Carbs 35 g 2.3 U

Bolus 2.3 U

Next



Menu

Suspend Delivery

Audio Options

History

Reservoir & Tubing

Insulin Settings



02

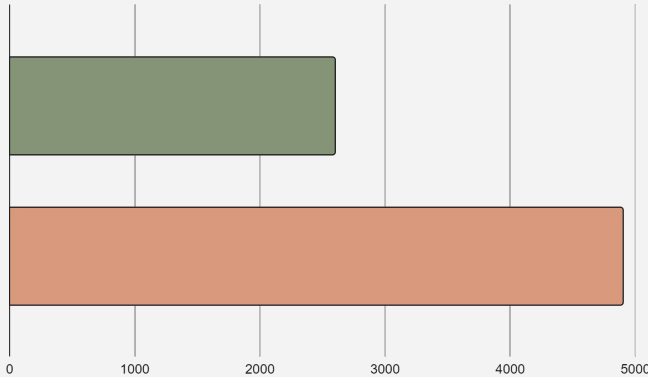
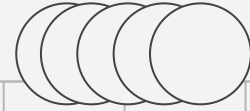
# Why try to replicate an insulin pump?

They already exist -  
what's the point?





# Cost



\$1,100 -  
\$2,600

Yearly  
Out-of-Pocket

Type 1 diabetes on  
multiple daily insulin  
injections

\$1,400 -  
\$4,900

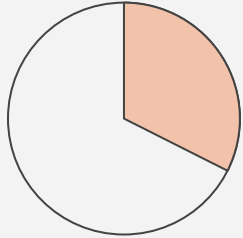
Yearly  
Out-of-Pocket

Type 1 diabetes on  
insulin pump therapy

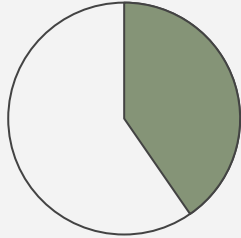
“A Medtronic insulin pump currently costs  
\$8,574” - Medtronic  
\$8,574 AUD → \$7,714 CAD

# Health

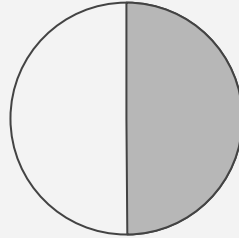
## Diabetes Contributes to...



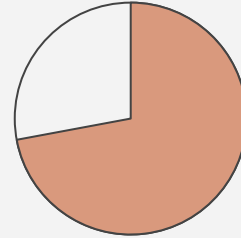
**30%**  
Strokes



**40%**  
Heart  
Attacks



**50%**  
Kidney  
Failure  
(requiring  
dialysis)



**70%**  
Non-traumatic  
leg & foot  
amputations

Reduces life  
span by 5-15  
years

More likely to  
be hospitalized  
for...

**3x:**  
Cardiovascular  
Disease

**12x:**  
end-stage  
renal disease

**20x:**  
non-traumatic  
lower limb  
amputation

# Considering Cost & Health...



01

## Cost Reduction

A standard insulin pump can cost approximately \$8,000 CAD, not including the supplies needed for insulin pump therapy.



02

## Accessible Healthcare

Lowering insulin pump costs makes it easier for diabetics to afford insulin pump therapy. Insurers and governments may also be more inclined to cover cheaper pumps.



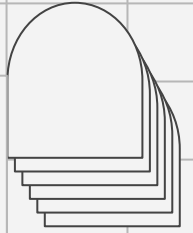
03

## Healthier Diabetics

Access to insulin pump therapy is unattainable for many. With access, diabetics can avoid unnecessary health complications.

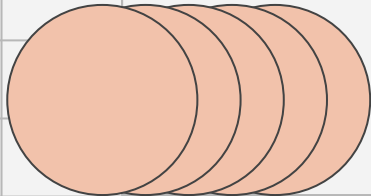


# Pens vs Pumps



Type 1 Diabetics can opt to use insulin pens: one pen for daily basal injections, and a second pen for bolus injections.

Using pens is not always convenient, and often yield different results for the patient. Unfortunately insulin pumps are not always covered by insurance or government programs, despite being a better option for many diabetics.



Insulin pumps are incredibly expensive, but should they be as expensive as they are, or could cheaper options be made available?



03

Code

Where all the magic  
happens!



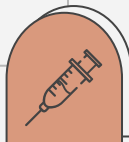


# Classes



## Basal Settings

Holds information about the basal patterns



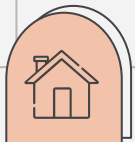
## Bolus Settings

Holds information about the bolus settings



## Menu

Holds all the different menus and their options



## UpdateThread

Displays necessary pump information



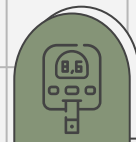
## BasalThread

Ensures basal insulin is delivered every minute



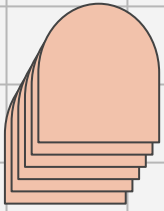
## ActiveInsulin Thread

Periodically reduces active insulin amount



## Pump

Holds settings, time, flags, and various other important pieces of information



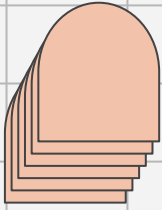
# Main

The Main.java class is responsible for only a few things:

- Reading the configs file
- Instantiating the pump
- Starting the BasalThread
- Starting the UpdateThread
- Starting the ActiveInsulinThread



```
public class Main {  
    public static Pump pump;  
  
    public static void main(String[] args) {  
        // create hashmaps based on config file  
        HashMap<String, ArrayList<String>> configs = init();  
  
        if(!configs.isEmpty()) {  
            // create pump with configs  
            pump = new Pump(configs);  
        } else {  
            // create pump with new configs  
            pump = new Pump();  
        }  
  
        // begin basal tasks  
        BasalThread basalThread = new BasalThread(pump);  
        Thread thread1 = new Thread(basalThread);  
        thread1.start();  
  
        // begin update tasks  
        UpdateThread updateThread = new UpdateThread(pump);  
        Thread thread2 = new Thread(updateThread);  
        thread2.start();  
  
        // begin activeInsulin tasks  
        ActiveInsulinThread activeInsulinThread = new ActiveInsulinThread(pump);  
        Thread thread3 = new Thread(activeInsulinThread);  
        thread3.start();  
    }  
}
```

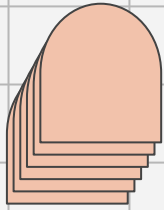


# Config File

- Reads “configs.txt” and stores information into a hashmap. This information is used to instantiate the Pump and its BolusSettings & BasalSettings classes.
- Current Basal Pattern is marked with an asterisk

1	CARB_RATIO	47	INSULIN_SENSITIVITY	48	BASAL_PATTERN 1 *
2	10.0	48	2.2	49	3.0
3	10.0	49	2.2	50	3.0
4	10.0	50	2.2	51	3.0
5	10.0	51	2.2	52	3.0
6	10.0	52	2.2	53	3.0
7	10.0	53	2.2	54	3.0
8	10.0	54	2.2	55	3.0
9	10.0	55	2.2	56	3.0
10	10.0	56	2.2	57	3.0
11	8.0	57	2.0	58	3.0
12	8.0	58	2.0	59	3.0
13	8.0	59	2.0	60	3.0
14	8.0	60	1.8	61	3.0
15	8.0	61	1.8	62	3.0
16	8.0	62	1.8	63	3.0
17	7.0	63	2.0	64	3.0
18	7.0	64	2.0	65	3.0
19	7.0	65	2.0	66	3.0
20	7.0	66	2.0	67	3.0
21	8.0	67	2.2	68	3.0
22	8.0	68	2.2	69	3.0
23	10.0	69	2.2	70	3.0
24	10.0	70	2.2	71	3.0
25	10.0	71	2.2	72	3.0
		72	2.0	73	3.0
		73	2.0	74	2.0
		74	2.0	75	2.0
		75	1.8	76	2.0
		76	1.8	77	2.0
		77	2.0	78	2.0
		78	2.0	79	2.0
		79	2.0	80	2.0
		80	2.2	81	2.0
		81	2.2	82	2.0
		82	2.2	83	2.0
		83	2.2	84	2.0
		84	2.0	85	2.0
		85	2.0	86	2.0
		86	2.2	87	2.0
		87	2.2	88	2.0
		88	2.2	89	2.0
		89	2.2	90	2.0
		90	2.2	91	2.0
		91	2.2	92	2.0
		92	2.2	93	2.0
		93	2.2	94	2.0
		94	2.2	95	2.0
		95	2.2	96	2.0
		96	2.2	97	2.0
		97	2.2	98	2.0
		98	2.2	99	2.0
		99	2.2	100	2.0
		100	2.2	101	2.0
		101	2.2	102	2.0
		102	2.2	103	2.0
		103	2.2	104	2.0
		104	2.2	105	2.0
		105	2.2	106	2.0
		106	2.2	107	2.0
		107	2.2	108	2.0
		108	2.2	109	2.0
		109	2.2	110	2.0
		110	2.2	111	2.0
		111	2.2	112	2.0
		112	2.2	113	2.0
		113	2.2	114	2.0
		114	2.2	115	2.0
		115	2.2	116	2.0
		116	2.2	117	2.0
		117	2.2	118	2.0
		118	2.2	119	2.0
		119	2.2	120	2.0
		120	2.2	121	2.0
		121	2.2	122	2.0
		122	2.2	123	2.0
		123	2.2	124	2.0
		124	2.2	125	2.0
		125	2.2	126	2.0
		126	2.2	127	2.0
		127	2.2	128	2.0
		128	2.2	129	2.0
		129	2.2	130	2.0
		130	2.2	131	2.0
		131	2.2	132	2.0
		132	2.2	133	2.0
		133	2.2	134	2.0
		134	2.2	135	2.0
		135	2.2	136	2.0
		136	2.2	137	2.0
		137	2.2	138	2.0
		138	2.2	139	2.0
		139	2.2	140	2.0
		140	2.2	141	2.0
		141	2.2	142	2.0
		142	2.2	143	2.0
		143	2.2	144	2.0
		144	2.2	145	2.0
		145	2.2	146	2.0
		146	2.2	147	2.0
		147	2.2	148	2.0
		148	2.2	149	2.0
		149	2.2	150	2.0
		150	2.2	151	2.0
		151	2.2	152	2.0
		152	2.2	153	2.0
		153	2.2	154	2.0
		154	2.2	155	2.0
		155	2.2	156	2.0
		156	2.2	157	2.0
		157	2.2	158	2.0
		158	2.2	159	2.0
		159	2.2	160	2.0
		160	2.2	161	2.0
		161	2.2	162	2.0
		162	2.2	163	2.0
		163	2.2	164	2.0
		164	2.2	165	2.0
		165	2.2	166	2.0
		166	2.2	167	2.0
		167	2.2	168	2.0
		168	2.2	169	2.0
		169	2.2	170	2.0
		170	2.2	171	2.0
		171	2.2	172	2.0
		172	2.2	173	2.0
		173	2.2	174	2.0
		174	2.2	175	2.0
		175	2.2	176	2.0
		176	2.2	177	2.0
		177	2.2	178	2.0
		178	2.2	179	2.0
		179	2.2	180	2.0
		180	2.2	181	2.0
		181	2.2	182	2.0
		182	2.2	183	2.0
		183	2.2	184	2.0
		184	2.2	185	2.0
		185	2.2	186	2.0
		186	2.2	187	2.0
		187	2.2	188	2.0
		188	2.2	189	2.0
		189	2.2	190	2.0
		190	2.2	191	2.0
		191	2.2	192	2.0
		192	2.2	193	2.0
		193	2.2	194	2.0
		194	2.2	195	2.0
		195	2.2	196	2.0
		196	2.2	197	2.0
		197	2.2	198	2.0
		198	2.2	199	2.0
		199	2.2	200	2.0
		200	2.2	201	2.0
		201	2.2	202	2.0
		202	2.2	203	2.0
		203	2.2	204	2.0
		204	2.2	205	2.0
		205	2.2	206	2.0
		206	2.2	207	2.0
		207	2.2	208	2.0
		208	2.2	209	2.0
		209	2.2	210	2.0
		210	2.2	211	2.0
		211	2.2	212	2.0
		212	2.2	213	2.0
		213	2.2	214	2.0
		214	2.2	215	2.0
		215	2.2	216	2.0
		216	2.2	217	2.0
		217	2.2	218	2.0
		218	2.2	219	2.0
		219	2.2	220	2.0
		220	2.2	221	2.0
		221	2.2	222	2.0
		222	2.2	223	2.0
		223	2.2	224	2.0
		224	2.2	225	2.0
		225	2.2	226	2.0
		226	2.2	227	2.0
		227	2.2	228	2.0
		228	2.2	229	2.0
		229	2.2	230	2.0
		230	2.2	231	2.0
		231	2.2	232	2.0
		232	2.2	233	2.0
		233	2.2	234	2.0
		234	2.2	235	2.0
		235	2.2	236	2.0
		236	2.2	237	2.0
		237	2.2	238	2.0
		238	2.2	239	2.0
		239	2.2	240	2.0
		240	2.2	241	2.0
		241	2.2	242	2.0
		242	2.2	243	2.0
		243	2.2	244	2.0
		244	2.2	245	2.0
		245	2.2	246	2.0
		246	2.2	247	2.0
		247	2.2	248	2.0
		248	2.2	249	2.0
		249	2.2	250	2.0
		250	2.2	251	2.0
		251	2.2	252	2.0
		252	2.2	253	2.0
		253	2.2	254	2.0
		254	2.2	255	2.0
		255	2.2	256	2.0
		256	2.2	257	2.0
		257	2.2	258	2.0
		258	2.2	259	2.0
		259	2.2	260	2.0
		260	2.2	261	2.0
		261	2.2	262	2.0
		262	2.2	263	2.0
		263	2.2	264	2.0
		264	2.2	265	2.0
		265	2.2	266	2.0
		266	2.2	267	2.0
		267	2.2	268	2.0
		268	2.2	269	2.0
		269	2.2	270	2.0
		270	2.2	271	2.0
		271	2.2	272	2.0
		272	2.2	273	2.0
		273	2.2	274	2.0
		274	2.2	275	2.0
		275	2.2	276	2.0
		276	2.2	277	2.0
		277	2.2	278	2.0
		278	2.2	279	2.0
		279	2.2	280	2.0
		280	2.2	281	2.0
		281	2.2	282	2.0
		282	2.2	283	2.0
		283	2.2	284	2.0
		284	2.2	285	2.0
		285	2.2	286	2.0
		286	2.2	287	2.0
		287	2.2	288	2.0
		288	2.2	289	2.0
		289	2.2	290	2.0
		290	2.2	291	2.0
		291	2.2	292	2.0
		292	2.2	293	2.0
		293	2.2	294	2.0
		294	2.2	295	2.0
		295	2.2	296	2.0
		296	2.2	297	2.0
		297	2.2	298	2.0
		298	2.2	299	2.0
		299	2.2	300	2.0
		300	2.2	301	2.0
		301	2.2	302	2.0
		302	2.2	303	2.0
		303	2.2	304	2.0
		304	2.2	305	2.0
		305	2.2	306	2.0
		306	2.2	307	2.0
		307	2.2	308	2.0
		308	2.2	309	2.0
		309	2.2	310	2.0
		310	2.2	311	2.0
		311	2.2	312	2.0
		312	2.2	313	2.0
		313	2.2	314	2.0
		314	2.2	315	2.0
		315	2.2	316	2.0
		316	2.2	317	2.0
		317	2.2	318	2.0
		318	2.2	319	2.0
		319	2.2	320	2.0
		320	2.2	321	2.0
		321	2.2		





# UpdateThread

Updates the console with important details, such as...

- Date & Time
- Active insulin
- Reservoir amount
- Prompt for menu input

This thread is paused if the user is within a menu.



```
package Classes;

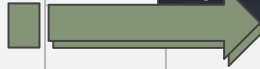
public class UpdateThread implements Runnable {

    private final Pump pump;
    private static final int ONE_MINUTE = 60000;
    private static final int TEN_SECONDS = 10000;

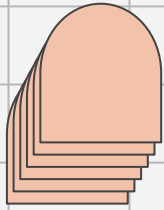
    public UpdateThread(Pump pump) { this.pump = pump; }

    @Override
    public void run() {
        // while the pump permits updates, print time, active insulin amount,
        // reservoir amount, and prompt for menu options.
        // Repeat as per variable selected above.
        while (Pump.update) {
            System.out.println("\n\n-----");
            System.out.println(pump.getTime());
            System.out.println("ACTIVE INSULIN:\t" + pump.getActiveInsulin());
            System.out.printf("RESERVOIR:\t\t%.3f\n", pump.getReservoir());
            System.out.println("OPTIONS:\t\t1.Bolus 2.Menu");
            pump.checkForMenu();

            try {
                Thread.sleep(TEN_SECONDS);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }
    }
}
```





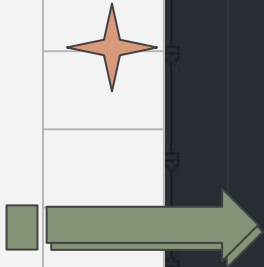
```
-----
Sun, 03 December, 2023 01:32 p.m.
ACTIVE INSULIN: 0.0
RESERVOIR:      86.967
OPTIONS:        1.Bolus 2.Menu
```



# BasalThread

Basal insulin should never be interrupted for any reason, unless the user wishes to suspend pump activity.

This thread checks if the pump is active every minute, and if so, will administer basal insulin.



```
package Classes;

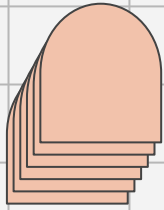
public class BasalThread implements Runnable {

    private final Pump pump;
    private static final int ONE_MINUTE = 60000;

    public BasalThread (Pump pump) {
        this.pump = pump;
    }

    @Override
    public void run() {
        // while the pump is active, run the basal() function
        // and repeat every 1 minute
        while (Pump.active && pump.getReservoir() > 0) {
            pump.basal();

            try {
                Thread.sleep(ONE_MINUTE);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }
    }
}
```



# ActiveInsulinThread

The active insulin is reduced depending on how many times the “ACTIVE\_INSULIN\_PER\_HOUR” variable.

This thread checks if the pump is “on” (which is always should be!). Even if the active insulin is not displayed (like on the UpdatedThread), it should still be reduced accordingly.



```
package Classes;

public class ActiveInsulinThread implements Runnable {

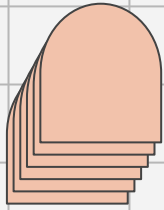
    private final Pump pump;
    private static final int ONE_HOUR = 3600000;
    private static final int UPDATE = ONE_HOUR / Pump.ACTIVE_INSULIN_PER_HOUR;

    public ActiveInsulinThread (Pump pump) { this.pump = pump; }

    @Override
    public void run() {
        // while the pump is turned on,
        // run the reduceActiveInsulin() function
        while (Pump.on) {
            pump.reduceActiveInsulin();

            try {
                Thread.sleep(UPDATE);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }
    }
}
```





# Pump Class

The Pump.java class is the largest class of all, as it includes...

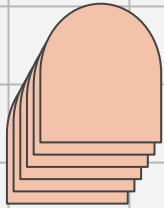
- Bolus & Basal settings
- Menus
- Date/time
- Several flags & warnings



```
public class Pump {
    protected static BolusSettings bolusSettings;
    protected static BasalSettings basalSettings;
    private static Menu menus = null;
    private Date time;
    public static boolean active = true;
    public static boolean update = true;
    public static boolean on = true;
    private boolean warning10 = false;
    private boolean warning20 = false;
    private boolean warning0 = false;
    private double reservoir;
    private ArrayList<Double> activeInsulin = new ArrayList<>();
    private static final int BASAL_PER_HOUR = 60;
    protected static final int ACTIVE_INSULIN_PER_HOUR = 60;
    private final SimpleDateFormat sdf = new SimpleDateFormat( pattern: "E, dd MMMM,
    yyyy hh:mm a");
    private final String warning0String = "WARNING: Reservoir has 0 units of
    insulin remaining. Please change the reservoir immediately.";
    public final static String configFilePath = "Configs/configs.txt";

    // ***** CONSTRUCTOR *****
    public Pump() {
        System.out.println("***NEW INSULIN PUMP***");
        setTime();
        System.out.println(sdf.format(time));
        bolusSettings = new BolusSettings();
        basalSettings = new BasalSettings();
        writeConfigs();
        newReservoir();
        menus = new Menu( pump: this);
    }

    public Pump(HashMap<String, ArrayList<String>> configs) {
        System.out.println("***WELCOME BACK***");
        setTime();
        bolusSettings = new BolusSettings(configs);
        basalSettings = new BasalSettings(configs);
        newReservoir();
        menus = new Menu( pump: this);
    }
}
```



# Examples

This is what some of the output looks like!

[Video Example Link](#)



```
-----  
Mon, 04 December, 2023 10:33 p.m.  
ACTIVE INSULIN: 10.00  
RESERVOIR:      76.97  
OPTIONS:        1.Bolus 2.Menu
```

```
-----  
Mon, 04 December, 2023 10:38 p.m.  
ACTIVE INSULIN: 9.17  
RESERVOIR:      76.80  
OPTIONS:        1.Bolus 2.Menu
```

```
-----  
Mon, 04 December, 2023 10:43 p.m.  
ACTIVE INSULIN: 8.33  
RESERVOIR:      76.63  
OPTIONS:        1.Bolus 2.Menu
```

## MAIN MENU:

1. Suspend/Resume Delivery
  2. New Reservoir
  3. Insulin Settings
- Q to exit

## INSULIN MENU:

1. Bolus Settings
  2. Basal Settings
- Q to exit

1

## BOLUS MENU:

1. Carb Ratio
  2. Insulin Sensitivity
  3. Insulin Longevity
  4. Target Glucose
- Q to exit

## BASAL MENU:

1. Add New Basal Pattern
  2. Delete Basal Pattern
  3. Select Current Basal Pattern
- Q to exit

2

Available Basal Patterns to delete:

Basal Pattern #1 - Daily Insulin of 70.0

Basal Pattern #2 - Daily Insulin of 61.0 \*



04

# Changes & Conclusions

What did I learn from  
this project?





## Language

**Console-Based:** C++

**Mobile App:** React Native



## Reading Input

**Did you know?**

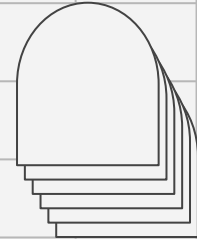
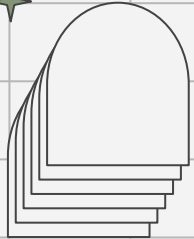
Scanners **block** threads! If there is no input, **there is no way to time it out.**

Next time, I would use BufferedReader and another thread to collect input.

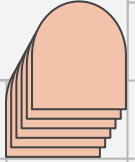
## Format

**Mobile App > Console**

- Closer to achieving Phase 2 of the project
- Less input parsing / validating (force certain keyboards for certain input types)
- No need to “update” the console after x seconds or minutes; the app would update in realtime

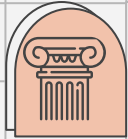


# Conclusions



## Cost

Insulin pumps **can** be made more affordable.



## Challenge

I learned a ton of new things through this project.



## Ease

The workings of an insulin pump aren't as complex as one would think & are easily replicable.

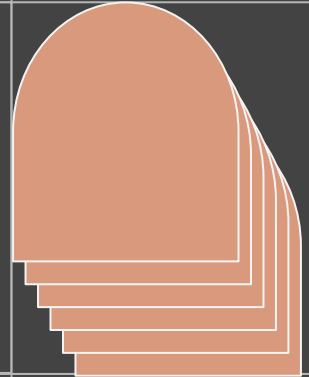
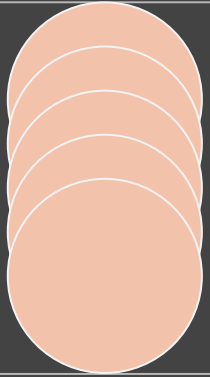
If one person can make the logic of the pump, it shouldn't take too large of a team to create a new pump design and bring it to life!





05

# Resources

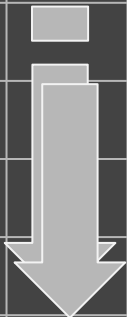


# Resource List



- **Diabetes Canada. (2022, February). Diabetes in Canada. Retrieved from Diabetes.ca:**  
[https://www.diabetes.ca/DiabetesCanadaWebsite/media/Advocacy-and-Policy/Backgrounder/2022\\_Backgrounder\\_Canada\\_English\\_1.pdf](https://www.diabetes.ca/DiabetesCanadaWebsite/media/Advocacy-and-Policy/Backgrounder/2022_Backgrounder_Canada_English_1.pdf)
- **John, C. w. (2021, June 28). Multithreading in Java Explained in 10 Minutes. Retrieved from YouTube.com:**  
[https://www.youtube.com/watch?v=r\\_MbozD32eo&ab\\_channel=CodingwithJohn](https://www.youtube.com/watch?v=r_MbozD32eo&ab_channel=CodingwithJohn)
- **Medtronic. (2023). Insulin Pump Costs. Retrieved from Medtronic-diabetes.com.au:**  
<https://hcp.medtronic-diabetes.com.au/insulin-pump-costs#:~:text=It%20is%20also%20important%20the,insulin%20pump%20currently%20costs%20%248%2C574>
- **Medtronic. (2023, 01 26). Minimed 630G System User Guide. Retrieved from medtronicdiabetes.com:**  
<https://www.medtronicdiabetes.com/sites/default/files/library/download-library/user-guides/MiniMed-630G-System-User-Guide.pdf>
- **Neverdal, C. (2016, March 8). How to overwrite an existing .txt file. Retrieved from StackOverflow.com:**  
<https://stackoverflow.com/questions/26785315/how-to-overwrite-an-existing-txt-file>
- **rohit2sahu. (2020, December 28). Reading Text File into Java HashMap. Retrieved from GeeksForGeeks.org:**  
<https://www.geeksforgeeks.org/reading-text-file-into-java-hashmap/>
- **Simpson, K. (2019, June 7). Java Library [#11] - Scheduling Tasks (Timer & TimerTask). Retrieved from Youtube.com:**  
[https://www.youtube.com/watch?v=tKSe8DAkrYk&ab\\_channel=KodySimpson](https://www.youtube.com/watch?v=tKSe8DAkrYk&ab_channel=KodySimpson)

Template: this presentation template was created by **SlidesGo**, and includes icons by **FlatIcon** and infographics & images by **Freepik**



# Thanks!

Any questions?

