Basic concepts

1.

a. Temporary Fields, sometimes when creating complex algorithm that need many input, programmer create fields for this data in class. But the problem is often the data only used in the algorithm and left unused rest of the time. This kind of code is tough to understand when we expect to see data in object fields but for some reason they are almost empty all the time. Temporary fields and all code operating on them can be put in a separate class, by creating a new class and place the fields and methods responsible for the relevant functionality in it, it will more easy to read and understand

b. Dependency injection is a programming technique that makes a class independent of its dependencies by passing dependency to other objects or framework(dependency injector), instead of recreating new object with new dependency each time we make a change we could just inject the dependency during runtime. Dependency injection can be usefull in unit testing and it will make change or extension of the program easier since we could always change the dependencies during runtime.

2. a. POST:

do: to send data to server

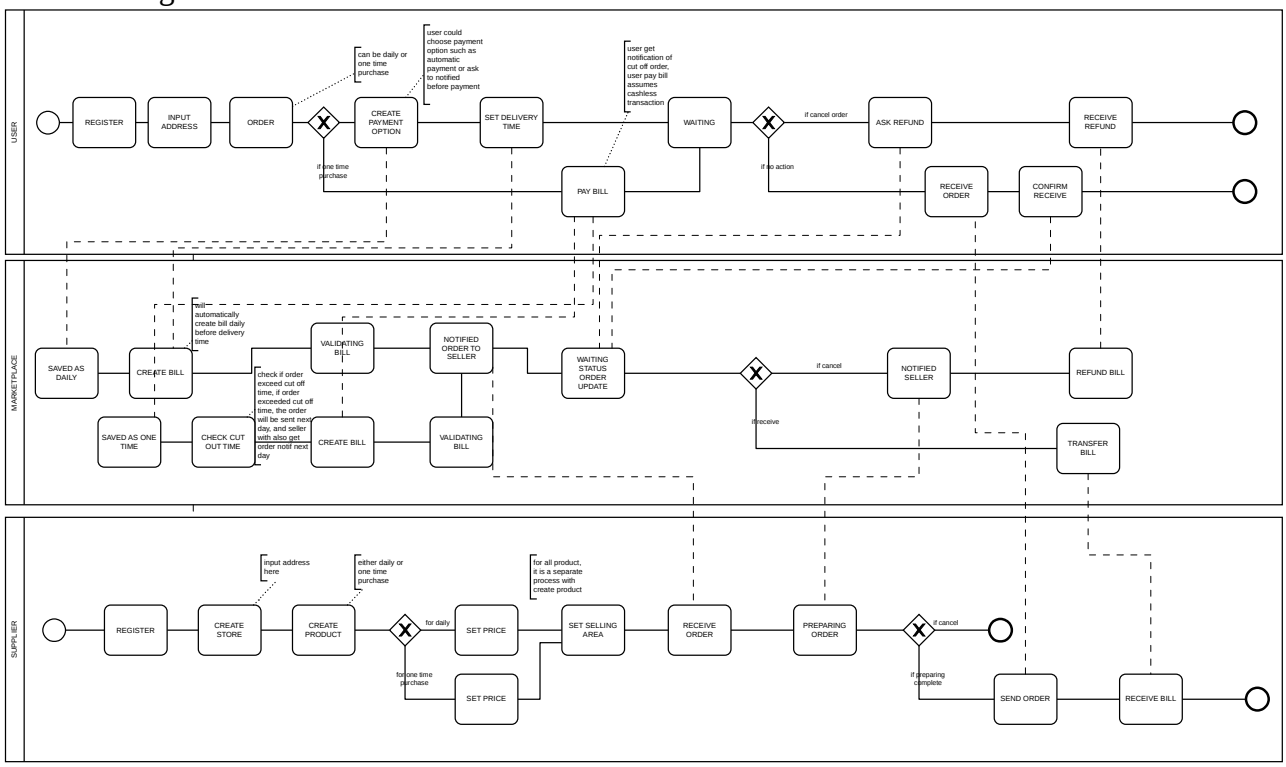
dont: to get data from server

b. Get:

do: to get data from server to display in client

dont: to post data, because all data will be visible

Basic Coding



Addition:

We can use Djikstra algoritm to decide shortes path.

PSEUDOCODE

```
function Dijkstra(Graph, source):

      create vertex set Q

      for each vertex v in Graph:
         dist[v] ← INFINITY
         prev[v] ← UNDEFINED
         add v to Q
     dist[source] ← 0

     while Q is not empty:
        u ← vertex in Q with min dist[u]

        remove u from Q

        for each neighbor v of u:          // only v that are still in Q
           alt ← dist[u] + length(u, v)
           if alt < dist[v]:
              dist[v] ← alt
              prev[v] ← u

     return dist[], prev[]
```

ALGORITM

1.

```
package main

import "fmt"

//input example
//6
//10 10 20 20 30 40

func main() {
        var n int
        fmt.Scan(&n)

        arr := make([]int, 101)

        count := 0

        for i := 0; i < n; i++ {
                var colorCode int
                fmt.Scan(&colorCode)

                arr[colorCode]++

                if (arr[colorCode]%2 == 0){
                count++
```

```
                }
        }

        fmt.Println(count)
 }
```

2.

```
package main

import (
    "fmt"
    "math"
    "os"
)

//input example
//10
//DUDUDUDUDU

func main() {
        var n int
      fmt.Scanln(&n)

        if (float64(n) > math.Pow(10,6)){
        fmt.Println("please input n no more than 1.000.000")
        os.Exit(1)
        }

        var arr string
        fmt.Scanln(&arr)

        arrLen := len(arr)
        if (arrLen!=n){
        fmt.Println("string input not match with n")
        os.Exit(1)
        }


        count := 0
        valley := 0
        valid := true

        hiking := make([]string, arrLen)
        for i:=1;i<=n;i++{
        var compare string = string(arr[i-1])
        if (compare=="D"){
                        hiking[i-1]=compare
        } else if(compare=="U"){
                        hiking[i-1]=compare
        } else{
                        valid = false
                        break
```

```
        }
        }

        if (valid){
        for i:=1;i<=n;i++{

                if(hiking[i-1]=="D"){
                        count--
                } else {
                        if((count+1)==0){

                        valley++
                }
                        count++
            }
       }

       fmt.Println(valley)
       } else {
       fmt.Println("wrong input")
    }

}
```

3.

```
func moneySeparator(string money){

        remove any character from string money (e.g "." "," "-" etc)
        count length of string, convert it to float64


        for (startIndex 1; index<=length of string; index++){
                get char from index[i-1] from string to var strNum
                parse the char to float64 to a var n

                if (no err){
                        print (n * 10 to the power of (length of string - index))
                }
        }

}
```

4.

```
package main

import (
        "fmt"
)

func main() {
```

```
	b:= make([]int, 100)

	for i := 1; i<=100; i++{
	   c:=100/i
	   m:=0
	   for j := 1; j<=c; j++{
	      if (m<=100){
	         m=m+i
	      }
	      if (b[m-1]==0) {
	         b[m-1]=1
	      } else {
	        b[m-1]=0
	      }
	   }

	}

	count := 0
	for _, v := range b {
		if (v == 1){
			count++
		}
	}
	fmt.Println(count)

}
```