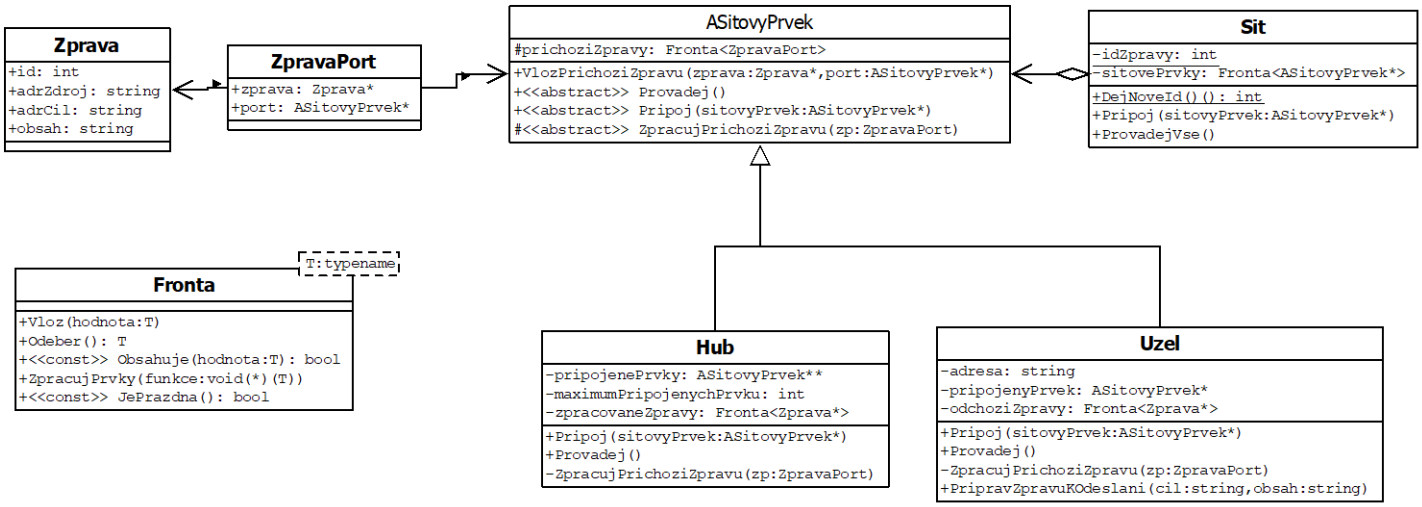
Cvičení 10

Vytvořte zjednodušený simulátor virtuální počítačové sítě.



Již realizované třídy:

- *Zprava* – paket
  - *id* – globální id zprávy v systému
  - *adrZdroj*, *adrCil* – adresa odesílatele a adresáta
  - *obsah* – text zprávy
- *ZpravaPort* – paket s asociovaným portem (zařízením, od kterého zpráva přišla)
- *Fronta* – kolekce prvků
  - *Vloz*, *Odeber*, *JePrazdna* – základní operace
  - *Obsahuje(T)* – vrací true, pokud je prvek ve frontě obsažen
  - *ZpracujPrvky(funkce)* – provede funkci nad každým prvkem ve frontě

Realizujte (detaily vizte UML diagram):

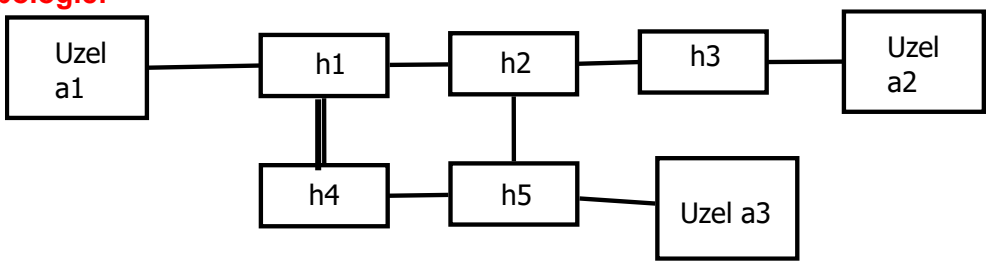
- *ASitovyPrvek* – abstraktní třída představující síťový prvek
  - *prichoziZpravy* – fronta příchozích zpráv
  - ---
  - *VlozPrichoziZpravu(zprava, port)* – vloží zprávu do atr. *prichoziZpravy*
  - *Provadej()* – abstraktní metoda, provede potřebné činnosti – dle typu zařízení
  - *Pripoj(sp)* – abstraktní konfigurační metoda – propojí aktuální prvek s parametrem (nastaví atribut, přidá do kolekce, …)
  - *ZpracujPrichoziZpravu(zp)* – zpracuje jednu příchozí zprávu

- *Hub* – síťový prvek Hub / Rozbočovač
  - *pripojenePrvky* – pole ukazatelů s připojenými prvky, alokován konstruktorem dle počtu portů, nepřipojené porty jsou *nullptr*
  - *maximumPripojenychPrvku* – velikost pole *pripojenePrvky*, nastaveno konstruktorem
  - *zpracovaneZpravy* – fronta, již zpracovaných zpráv (pro zabránění dvojího zpracování zpráv)
  - ---
  - *Pripoj()* – propojí prvky, přidá odkaz na prvek do pole *pripojenePrvky*
  - *Provadej()* – pro každou zprávu ve frontě příchozích zpráv vyvolá metodu *ZpracujPrichoziZpravu*
  - *ZpracujPrichoziZpravu(zp)*
    - Pokud byla zpráva v minulosti již zpracována → konec zpracování.
    - Rozešli zprávu všem připojeným prvkům (pomocí *VlozPrichoziZpravu*), kromě prvku, ze kterého byla zpráva přijata.
    - Zařaď zprávu do fronty již zpracovaných zpráv.

- *Uzel* – síťový prvek koncová stanice
  - *Adresa* – adresa aktuálního uzlu
  - *pripojenyPrvek* – připojený síťový prvek k uzlu (přes tento prvek jsou odesílány zprávy)
  - *odchoziZpravy* – fronta zpráv připravených k odeslání
  - ---
  - *Pripoj()* – propojí prvky, nastaví odkaz do atr. *pripojenyPrvek*
  - *Provadej()*
    - Všechny zprávy ve frontě *odchoziZpravy* odešle přes *pripojenyPrvek* (*VlozPrichoziZpravu*)
    - Pro všechny zprávy ve frontě *prichoziZpravy* vyvolá *ZpracujPrichoziZpravu*
  - *ZpracujPrichoziZpravu*
    - Zkontroluje, jestli je zprava určena pro aktuální uzel, pokud ne → konec zpracování.
    - Vypíše informaci o příchozí zprávě se všemi detaily.
    - Pokud zpráva obsahuje text „**ping**“
      - Připraví k odeslání zprávu s textem „**pong**“ odesílateli zprávy.
  - *PripravZpravuKOdeslani*() – přidá zprávu do fronty odchozích zpráv

- *Sit* – řídící struktura pro celou síť
  - *idZpravy* – statický čítač pro počítání zpráv
  - *sitovePrvky* – všechny síťové prvky v síti
  - ---
  - *DejNoveId()* – vrátí nové id zprávy
  - *Pripoj()* – přidá síťový prvek do fronty *sitovePrvky* (pouze pokud již ve frontě není obsažen)
  - *ProvadejVse()*
    - Vyvolá metodu *Provadej()* nad každým prvek v síti pomocí metody fronta – *ZpracujPrvky*. Pro realizaci vytvořte pomocnou privátní statickou metodu nebo lambda výraz.

Připravená topologie:



Očekávaný výstup:

```
a2 RECV id:0 src:a1 msg:ping
a3 RECV id:1 src:a1 msg:ping
a1 RECV id:2 src:a2 msg:pong
a1 RECV id:3 src:a3 msg:pong
```

Připravené třídy a main:

https://pastebin.com/k5Tgmum2

Cvičení odevzdejte na STAG – cvičení 10

Speciální bonusové body:

- Vytvořte síťový prvek přepínač (switch); pokud zná přes který port je adresát dostupný – odesílá jen tam, jinak se chová jako hub; přepínač na počátku nezná mapování – učí se procházejícími daty
- Aby topologie fungovala bez úprav i se switchi – naučte switche zjednodušenou obdobu STP protokolu; STP musí být založen na předávání zpráv; switch má definovanou prioritu (uvažujte, že pouze jeden switch bude mít nejvyšší prioritu); jako metriku využijte „hop count“ a vybudujte strom směrem ke kořenovému přepínači (s nejvyšší prioritou); algoritmus nemusí disponovat mechanismem opakovaného přepočtu po uplynutí časové periody – stačí pouze inicializační výpočet; dokud není stp konvergované – switch nepřeposílá data; pro simulaci můžete uvažovat dobu náběhu (x volání Provadej), aby bylo dosaženo STP konvergence.

```
#include <string>
#include <iostream>

using namespace std;

struct Zprava {
	int id;
	string adrZdroj;
	string adrCil;
	string obsah;

	Zprava() { }
	Zprava(int id, string adrZdroj, string adrCil, string obsah) :
		id(id), adrZdroj(adrZdroj), adrCil(adrCil), obsah(obsah) {
	}
};

struct ASitovyPrvek;

struct ZpravaPort {
	Zprava* zprava;
	ASitovyPrvek* port;

	ZpravaPort() { }
	ZpravaPort(Zprava* zprava, ASitovyPrvek* port) : zprava(zprava), port(port) {
	}
};

template <typename T>
struct Fronta {
private:
	struct El {
		T hodnota;
		El* dalsi;
		El* predchozi;
	};

	El* prvni;
	El* posledni;
public:
	Fronta() {
		prvni = posledni = nullptr;
	}

	Fronta(const Fronta& f) {
		El* it = f.prvni;
		while (it) {
			Vloz(it->hodnota);
			it = it->dalsi;
		}
	}

	~Fronta() {
		while (prvni) {
			El* tmp = prvni;
			prvni = prvni->dalsi;
			delete tmp;
		}
	}

	void Vloz(T hodnota) {
		posledni = new El{ hodnota, nullptr, posledni };
		if (!prvni)
			prvni = posledni;
		else
			posledni->predchozi->dalsi = posledni;
	}

	T Odeber() {
```

```
		T hodnota = posledni->hodnota;
		El* tmp = posledni;

		if (posledni->predchozi) {
			posledni = posledni->predchozi;
			posledni->dalsi = nullptr;
		}
		else {
			posledni = nullptr;
			prvni = nullptr;
		}

		delete tmp;
		return hodnota;
	}

	bool Obsahuje(T hodnota) const {
		El* el = prvni;
		while (el) {
			if (el->hodnota == hodnota)
				return true;

			el = el->dalsi;
		}

		return false;
	}

	using ApplyFunkce = void(*) (T);

	void ZpracujPrvky(ApplyFunkce f) {
		El* el = prvni;
		while (el) {
			f(el->hodnota);

			el = el->dalsi;
		}
	}

	bool JePrazdna() const {
		return prvni == nullptr;
	}
};


////////////////////////////////////////////////////////////////////////////////////////////////////////////////


int main()
{
	Sit* s = new Sit{ };

	Hub* h1 = new Hub{ 8 };
	Hub* h2 = new Hub{ 8 };
	Hub* h3 = new Hub{ 8 };
	Hub* h4 = new Hub{ 16 };
	Hub* h5 = new Hub{ 24 };

	Uzel* u1 = new Uzel{ "a1" };
	Uzel* u2 = new Uzel{ "a2" };
	Uzel* u3 = new Uzel{ "a3" };

#define propoj(a,b) a->Pripoj(b); b->Pripoj(a)
#define propojs(a,b,c) propoj(a,b); c->Pripoj(a); c->Pripoj(b)

	propojs(u1, h1, s);
	propojs(u2, h3, s);
	propojs(u3, h5, s);
```

```
        propojs(h1, h2, s);
        propojs(h2, h3, s);
        propojs(h1, h4, s);
        propojs(h4, h5, s);
        propojs(h5, h2, s);

        u1->PripravZpravuKOdeslani("a2", "ping");
        u1->PripravZpravuKOdeslani("a3", "ping");

        for (int i = 0; i < 100; i++)
               s->ProvadejVse();

    return 0;
}
```