**Názov: Tvorba komixu**

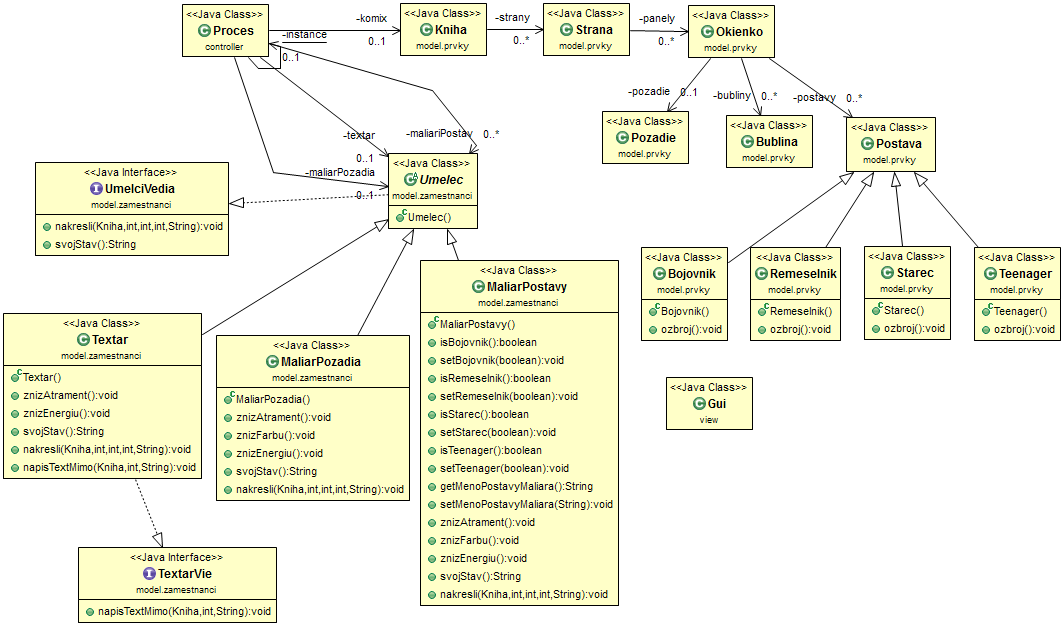
**Autor: Ivan Gulis**

Záznam môžeme chápať ako umelecké dielo, na ktorom spolupracujú viacerí umelci. Každý je v niečom dobrý, každý má iné znalosti či zručnosti. Hudobníci, architekti, maliari či spisovatelia si občas vystačia sami, ale niekedy je nutné do tvorby umeleckého diela zapojiť väčší kolektív.

Komix, spojenie textu, obrazu a deja, je v dnešnom storočí veľmi populárny v Amerike a v Japonsku(kde sa nazýva manga). Na jeho vytvorenie je potrebné vymyslieť dej tak, aby čitateľa zaujal. K pozitívnemu prvému dojmu dopomáha bohato ilustrovaný obal knižky či časopisu. V komixe ďalej vystupujú postavy(často ručne kreslené), pestré pozadie a iné objekty. Stránky sú členené na menšie panely hranatých tvarov, obklopené autorskými poznámkami všade po okrajoch. Všetko dopĺňajú bubliny s textom, najčastejšie elipsovité alebo v tvare mnohouholníkov. Ešte treba dodať, že japonská manga je celá čierno-biela(okrem obalu knihy), a máva menší počet panelov na strane(občas celá strana jeden obraz).

Je teda potrebné zabezpečiť človeka na napísanie deja. Ďalej maliara, ktorý celý komix(ak to nie je manga) vyfarbí na počítači alebo ručne. Každú postavu zvlášť zvykne kresliť jeden človek v celej knižke, preto platí pravidlo: ,,Koľko postáv, toľko kresličov." Pozadie, hlavne obloha, sa už tradične maľuje štetcom, aby postavy viac vynikli. Nakoniec je potrebné pridať bubliny s textom tak, aby nezakrývali príliš veľkú časť obrazu(ak je to nutné, je možné zmenšiť počet panelov na stránke, pričom sa obraz zväčší, čo je jedným zo znakov mangy).

A nesmieme zabudnúť na človeka, ktorý má na starosti spojenie všetkých spomenutých útržkov, vytvorených kolaboratívnou tvorbou umelcov. Tento človek rozhoduje o kompozícii každej vznikajúcej stránky a svojím spôsobom riadi prácu ostatných.

Výsledkom takejto tvorby bude záznam, komixová kniha, zložená z menších komponentov(postavy, pozadie, farby, text, bubliny, panely) rôzne umiestnených na každej stránke.

Na diagrame je hierarchia Umelca, dedia od nej Textar, MaliarPozadia a MaliarPostavy. Textár implementuje rozhranie TextarVie s metódou napisTextMimo(). Umelec implementuje rozhranie UmelciVedia s metódami svojStav() a nakresli(), ktoré prekonávajú dediči. Druhá hierarchia je nadradenej triedy Postava, a dedia od nej Bojovnik, Remeselnik, Starec a Teenager, prekonávajú metódu ozbroj(). Trieda Proces je hlavná trieda, singleton s jedinou instanciou, a ako atribúty má objekty tried Textar, MaliarPozadia, a ArrayList objektov triedy MaliarPostavy. Ďalej obsahuje objekt triedy Kniha, ktorá obsahuje ako atribút ArrayList objektov triedy Strana, ktorá obsahuje ArrayList triedy Okienko. Okienko obsahuje ako atribúty ArrayListy objektov prvkov Bublina a Postava.

**Predbežné verzie projektu:**

Prvá verzia vznikla na 7. cvičení 1.4.2014.

Vznikla hlavná trieda Proces, a prvá hierarchia zamestnancov. Pridal som im atribúty, a funkciu nakresli(), kde som prvý krát použil polymorfizmus aspoň na rozdielny výpis. Jednalo sa o objekty MaliaraPostavy a MaliaraPozadia. Najvyššia trieda Clovek bola neskôr zrušená.

Druhá verzia vznikla na 8. cvičení 8.1.2014.

V tejto verzii som rozšíril počet tried, a rozdelil ich ako prvky záznamu(bublina, okno, postava, pozadie, text) a zamestnancov som ďalej rozdelil na spisovatelia a maliari. Neskôr boli zaradení ako umelci. Maliarom som vytvoril metódy na znižovanie/zvyšovanie atramentu, energie a farby.

Tretia verzia bola Pracovná verzia programu.

Zmien pribudlo neuveriteľne veľa. Balíky som rozdelil do Model-View-Controller.

Controller pozostával z hlavnej triedy Proces, v ktorej sa okrem načítania Gui nachádzali všetci zamestnanci ako objekty: textár, maliar pozadia a ArrayList maliarov postáv. Ďalej veľké množstvo funkcii, ktoré volalo moje GUI(trieda v balíku View). Zhruba 70 komponentov tlačidiel, text fieldov a okien. ActionListenery ako anonymné triedy. Metódy z triedy Proces sa volali vytvoreným objektom typu Proces v triede Gui. Hned v úvodnom okne sa dalo vybrať, za akého zamestnanca chceme pracovať, riaditeľ, textár, maliar pozadia či postavy, a tlačidlo na pár výpisov.

Balík model, a jeho podbalíky prvky a zamestnanci obsahovali všetky zvyšné triedy. Základná hierarchia bola u zamestnancov: najvyššie bola trieda Umelec, ktorá obsahovala všetky potrebné triedy pre dedenie a atribúty. Túto triedu ďalej rozširovali triedy Textár, MaliarPostavy a MaliarPozadia. Textár mal viac atramentu a vedel písať text mimo bublín, maliari mali na viac atribút farbu. Všetci dedili rovnaké metódy nakresli() a zistiStav(). V oboch bol využitý polymorfizmus, naždý kreslil to svoje, prečo bol zamestnaný, a zisti stav ukázal aktuálny stav ich zdrojov (atribútov).

Druhý podbalík prvky pozostával z knihy, ktorá obsahovala List stránok, ktoré obsahovali list okienok. Každé okienko obsahovalo ďalej listy postáv a bublín s textom. Zamestnanci dopĺňali ich atribúty a tvorili ich objekty. Záznam ako kniha bola jedna, komix vytvorený v hlavnej triede Proces. Takto vznikla agregácia prvkov.

Štvrtá verzia vznikla na 10. cvičení 6.5.2014.

Z triedy proces som vytvoril Singletona, jedináčika, a tak som sa vyhol tvorbe objektu Proces v Gui a zároveň použil návrhový vzor. Ďalej som sa pokúšal o serializáciu, ktorá nedopadla úspešne.

Piata, posledná a súčastná verzia

V Gui sa toho veľa nezmenilo, drobné úpravy v rozmiestnení. Trieda proces získala pár nových metód. V tejto verzii sa menili, rozširovali a tvorili hierarchie. Vytvoril som dve rozhrania UmelciVedia a TextarVie, a tak sa vyhol nežiaducemu dedeniu zbytočných metód. Atribúty Umelca sú teraz typu final. Ako druhú hierarchiu som rozšíril triedu Postava. Dediči sú Bojovník, Remeselník, Starec a Teenager, všetci majú iné atribúty, a dedia metódu Ozbroj. Každý z nich má po ruke inú zbraň, a nie všetci sú s ňou bojaschopný. Pri maľovaní postavy si maliar vyberie, či chce postavu ozbrojenú alebo nie. Ďalej si môže vybrať zo 4 povolaní. Takto vznikol polymorfizmus aj v druhej hierarchii. K tomu som vytvoril Javadoc a diagram.

**Využitie prvkov javy:**

**Dedenie**

V programe sa nachádzajú 2 hierarchie, umelci a postavičky.

Nadradená trieda je Umelec (abstraktná), a dedia od nej Textar, MaliarPozadia a MaliarPostavy. Dalej rozhranie UmelciVedia implementuje triedu Umelec, a sú v nej metódy pre polymorfizmus. Rozhranie TextarVie obsahuje metódu napisTextMino(), ktorú vie vykonávať iba objekt tejto triedy.

abstract public class Umelec implements UmelciVedia {...}

public class Textar extends Umelec implements TextarVie {…}

public class MaliarPozadia extends Umelec {…}

public class MaliarPostavy extends Umelec {…}

public interface UmelciVedia {

public void nakresli(Kniha komix, int z, int x, int y, String s);

public String svojStav();

}

public interface TextarVie {

public void napisTextMimo(Kniha komix, int stranaCislo, String s);

}

Druhá hierarchia dedenia je tvorená postavičkami.

Nadradená trieda Postava, a dedia od nej Bojovník, Remeselník, Starec a Teenager. Každý má iné hodnoty atribútov, a dedia metódu Ozbroj. Autor si môže vybrať, či chce postavičku ozbrojenú, aké povolanie, a podľa toho sa postavička ozbrojí zbraňou, ktorú má po ruke (ktorú vlastní).

public class Postava {

protected String meno;

protected int vekPriemerne;

protected String vzdelanie;

protected String výzbroj = "nič";

protected boolean bojaschopný;

public void ozbroj() {}

}

public class Bojovnik extends Postava {…}

public class Remeselnik extends Postava {…}

public class Starec extends Postava {…}

public class Teenager extends Postava {…}

**Polymorfizmus:**

Vo vyššie spomenutej hierarchii Umelcove prázdne metódy nakresli() a svojStav() z implementovaného interface ďalej prekonávajú jeho dediči. Vo funkcii svojStav() sa vypíše aktuálny stav atribútov, funkcia nakresli() ich u každého znižuje inak, respektíve každý dedič má iné hodnoty vo funkcii znizAtrament(),energiu,farbu (textár atribút farba nemá). Funkcie svojStav() a nakresli() následne volám v Gui cez ActionListenera.

**public** **class** Proces {

**private** Umelec textar = **new** Textar();

**private** Umelec maliarPozadia = **new** MaliarPozadia();

**private** List<Umelec> maliariPostav = **new** ArrayList<>();

...}

**public** **interface** UmelciVedia {

**public** **void** nakresli(Kniha komix, **int** z, **int** x, **int** y, String s);

**public** String svojStav();

}

**abstract** **public** **class** Umelec **implements** UmelciVedia {

**protected** **int** energia = 100;

**protected** **int** atrament = 5000;

/\*\* Hodnoty spotreby pre kreslené objekty \*/

**protected** **final** **int** spotrebaAtramentuText = 250;

**protected** **final** **int** spotrebaAtramentuPostava = 125;

**protected** **final** **int** spotrebaAtramentuPozadie = 250;

**protected** **final** **int** spotrebaFarbyPostava = 250;

**protected** **final** **int** spotrebaFarbyPozadie = 500;

**protected** **final** **int** spotrebaEnergiePostava = 2;

**protected** **final** **int** spotrebaEnergiePozadie = 4;

**protected** **final** **int** spotrebaEnergieText = 1;}

**public** **class** Textar **extends** Umelec **implements** TextarVie {

/\*\* Vypíše svoj stav atribútov atrament a energia \*/

**public** String svojStav() {

**return** ("Zostáva mi už len " + **this**.atrament + " atramentu a "

+ **this**.energia + " energie." + "\n");

}

/\*\* Každý umelec kreslí niečo iné, tento píše text do bublín \*/

**public** **void** nakresli(Kniha komix, **int** stranaCislo, **int** panelCislo,

**int** bublinaCislo, String s) {

komix.getStrany().get(stranaCislo - 1).getPanely()[panelCislo - 1]

.getBubliny()[bublinaCislo - 1] = **new** Bublina();

komix.getStrany().get(stranaCislo - 1).getPanely()[panelCislo - 1]

.getBubliny()[bublinaCislo - 1].setText(s);

znizAtrament();

znizEnergiu();

}

}

**public** **class** MaliarPozadia **extends** Umelec {

/\*\* Vypíše svoj stav atribútov farba, atrament a energia \*/

**public** String svojStav() {

**return** ("Zostáva mi už len " + **this**.atrament + " atramentu, "

+ **this**.farba + " farby a " + **this**.energia + " energie." + "\n");

}

/\*\* Každý umelec kreslí niečo iné, tento nakreslí nejaké pozadie \*/

**public** **void** nakresli(Kniha komix, **int** stranaCislo, **int** panelCislo, **int** i,

String s) {

komix.getStrany().get(stranaCislo - 1).getPanely()[panelCislo - 1]

.getPozadie().setNazov(s);

znizAtrament();

znizFarbu();

znizEnergiu();

}

}

**public** **class** MaliarPostavy **extends** Umelec {

/\*\* Každý maliar postavy kreslí iba 1 postavičku \*/

**private** String menoPostavyMaliara;

**private** **boolean** bojovnik = **false**;

**private** **boolean** remeselnik = **false**;

**private** **boolean** starec = **false**;

**private** **boolean** teenager = **false**;

/\*\* Vypíše svoj stav atribútov farba, atrament a energia \*/

**public** String svojStav() {

**return** ("Zostáva mi už len " + **this**.atrament + " atramentu, "

+ **this**.farba + " farby a " + **this**.energia + " energie." + "\n");

}

/\*\* Každý umelec kreslí niečo iné, tento nakreslí svoju postavičku \*/

**public** **void** nakresli(Kniha komix, **int** stranaCislo, **int** panelCislo,

**int** postavaCislo, String s) {

/\*\* Nakreslí bojovníka \*/

**if** (**this**.bojovnik)

komix.getStrany().get(stranaCislo - 1).getPanely()[panelCislo - 1]

.getPostavy()[postavaCislo - 1] = **new** Bojovnik();

/\*\* Nakreslí remeselníka \*/

**if** (**this**.remeselnik)

komix.getStrany().get(stranaCislo - 1).getPanely()[panelCislo - 1]

.getPostavy()[postavaCislo - 1] = **new** Remeselnik();

/\*\* Nakreslí starca \*/

**if** (**this**.starec)

komix.getStrany().get(stranaCislo - 1).getPanely()[panelCislo - 1]

.getPostavy()[postavaCislo - 1] = **new** Starec();

/\*\* Nakreslí teenagera \*/

**if** (**this**.teenager)

komix.getStrany().get(stranaCislo - 1).getPanely()[panelCislo - 1]

.getPostavy()[postavaCislo - 1] = **new** Teenager();

/\*\* Pomenuje ho \*/

komix.getStrany().get(stranaCislo - 1).getPanely()[panelCislo - 1]

.getPostavy()[postavaCislo - 1]

.setMeno(**this**.menoPostavyMaliara);

znizAtrament();

znizFarbu();

znizEnergiu();

}

}

**Druhý polymorfizmus je použitý v druhej hierarchii, triedy Postava:**

**public** **class** Postava {

**protected** String meno;

**protected** **int** vekPriemerne;

**protected** String vzdelanie;

**protected** String výzbroj = "nič";

**protected** **boolean** bojaschopný;

**public** **void** ozbroj() {

}

**public** String getMeno() {

**return** meno;

}

**public** **void** setMeno(String meno) {

**this**.meno = meno;

}

**public** String getVýzbroj() {

**return** výzbroj;

}

**public** **void** setVýzbroj(String výzbroj) {

**this**.výzbroj = výzbroj;

}

}

**public** **class** Bojovnik **extends** Postava {

**public** Bojovnik() {

**this**.vekPriemerne = 30;

**this**.vzdelanie = "nevzdelaný";

**this**.bojaschopný = **false**;

}

**public** **void** ozbroj() {

**this**.výzbroj = "katana";

**this**.bojaschopný = **true**;

}

}

**public** **class** Remeselnik **extends** Postava {

**public** Remeselnik() {

**this**.vekPriemerne = 40;

**this**.vzdelanie = "vyučený";

**this**.bojaschopný = **false**;

}

**public** **void** ozbroj() {

**this**.výzbroj = "kladivo";

**this**.bojaschopný = **true**;

}

}

A podobne starec a teenager.

**Agregácia:**

Celý môj záznam ako komix je tvorený agregáciou. Všetko to začína v triede Proces.

**public** **class** Proces {

**private** Kniha komix = **new** Kniha();...}

**public** **class** Kniha {

**private** List<Strana> strany = **new** ArrayList<Strana>();...}

**public** **class** Strana {

**private** Okienko panely[];...}

**private** Bublina[] bubliny;

**private** Postava[] postavy;

**private** Pozadie pozadie = **new** Pozadie();}

**public** **class** Bublina {

**private** String text;...}

**public** **class** Postava {...}

**public** **class** Pozadie {

**private** String nazov;...}

**Zapúzdrenie:**

Zapúzdrenia som použil skutočne dostatok, ako vo vyššie uvedených kusoch prigramu. Každý private atribút v agregovaných triedach Knihy má tiež gettery/settery.

Uvediem len pár príkladov:

Metóda pocetVsetkychStran(), ktorá zistí počet strán. Je možné ju správne vykonať práve vďaka použitiu zapúzdrenia atribútov v triede Kniha.

**public** **class** Proces {...

**public** String pocetVsetkychStran() {

**return** ("Počet všetkych stran komixu je: "

+ **this**.komix.getStrany().size() + "\n");

...}

**public** **class** Kniha {

**private** List<Strana> strany = **new** ArrayList<Strana>();

**public** List<Strana> getStrany() {

**return** strany;

}

**public** **void** setStrany(List<Strana> strany) {

**this**.strany = strany;

}

}

V nasledujúcej metóde ozbrojenaPostava() je vidieť použitie metódy ozbroj(), ktorá bola vyššie popísaná bližšie pre polymorfizmus hierarchie triedy Postava. Túto metódu je možné vykonať práve vďaka zapúzdreniu a volaniu atribútov cez getter .get v tejto širokej agregácii.

**public** **void** ozbrojenaPostava(**int** stranaCislo, **int** panelCislo,

**int** postavaCislo) {

**this**.komix.getStrany().get(stranaCislo - 1).getPanely()[panelCislo - 1]

.getPostavy()[postavaCislo - 1].ozbroj();

}

**public** **class** Okienko {

**private** Postava[] postavy;

**public** Postava[] getPostavy() {

**return** postavy;

}

**public** **void** setPostavy(Postava[] postavy) {

**this**.postavy = postavy;

}

**Ďalšie:**

Pre návrhový vzor som využil singletona, jedináčika, a to pre metódu proces:

**private** **static** Proces *instance* = **null**;

**private** Proces() {

}

**public** **static** Proces getInstance() {

**if** (*instance* == **null**)

*instance* = **new** Proces();

**return** *instance*;

}

Následne volám jej inštanciu v Gui. Napríklad pre zavolanie už spomenutej metódy pocetVsetkychStran().

/\*\* Vypíše počet vytvorených strán \*/

bPočetVsetkychStran.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

vypisVsetkehoDruhu.append(Proces.*getInstance*()

.pocetVsetkychStran());

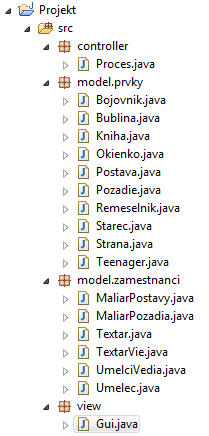
}

});

Tiež je možné si na tomto príklade všimnúť využitie anonymných tried pri volaní ActionListenerov.

**Balíky:**

Použil som tiež vhodnú organizáciu balíkov. Vzor Model-View-Controller. Kde model má 2 podbalíky.



**Manuál:**

Hneď ako spustíte program, objaví sa Hlavné Okno. Ste postavený do role riaditeľa projektu na tvorbu komixu. Rozhodujete o kompozícii, a celý komix je vo vašich rukách.

**Riaditeľ:**

Začnete kliknutím na tlačidlo Riaditeľ. Zadáte názov komixu podľa vašich predstáv, *napríklad "Guliverove cesty",* a názov tlačidlom uložíte.

Odomkne sa vám možnosť pridávať nové strany. Zadáte číslo, koľko okienok bude mat vaša prvá strana, a uložíte.

Nové strany pridávate opätovným stlačením tlačidla **Uložiť počet okienok** a môžete zakaždým zmeniť počet okienok.

V textovom okne vidíte čo ste vytvorili. Ďalej v tomto okne viete dopredu naplánovať počet postáv a bublín s textom v určitom okienku. *Napr. do prvej strany, do prvého okienka nastavíte, že tam budú dve postavy, a 2 bubliny (pekný dialóg dvoch osôb).*

Pozor, pridávať môžete iba do strán či okienok, ktoré ste už vytvorili!

O celej kompozícii komixu rozhodujete tu. Zatiaľ je ale okrem názvu prázdny.

Až ste skončili s vytvorením kompozície, môžete pokračovať v pôvodnom okne, a prihlásiť sa ako zamestnanec textár či maliar pozadia, alebo si najať potrebný počet maliarov postáv. Každý sa obmedzenú zásobu atramentu či farieb, a môžu sa časom vyčerpať. Komix nemáva veľa strán, preto nepotrebuje toľko zamestnancov.

**Textár:**

Ako textár máte na výber, kam chcete svoj text písať, a tvoriť tak dej. Do bubliny alebo mimo okienok, kam sa často píšu autorské komentáre či vysvetlivky.

Vyberiete tlačidlo **Vypĺňaj bubliny**.

Do prvého riadka napíšete váš text, *napr. "Ahoj, ako sa máš?".*

Potom nastavíte do ktorej bubliny sa má text priradiť, napr. do prvej strany, prvého okienka, do prvej bubliny.

Pozor, opäť pridávať môžete iba do strán, okienok či bublín, ktoré ste už vytvorili!

Do druhej bubliny môžete pridať napríklad *odpoveď "Mám sa dobre, ďakujem.",* a takto rozvinúť tento dialóg dvoch osôb spomenutý v príklade vyššie. Tiež máte možnosť ovplyvniť, či táto bublina má byť hranatá zaškrknutím políčka. Stlačením tlačidla **Ulož bublinu**, bublinu uložíte.

Späť sa dostanete tlačidlom Vrátiť sa do výberu.

Teraz môžete vybrať tlačidlo **Vypĺňaj text mimo okienok.**

Sem môžete, podobne ako v predchádzajúcom tlačidle, napísať text *(napr. "Tu prebieha rozhovor dvoch priateľov.")*. Text uložíte tlačidlom **Ulož tento text.**

Pozor, opäť pridávať môžete iba do strán, ktoré ste už vytvorili!

**Maliar pozadia:**

Ako maliar pozadia môžete do okienok na stranách pridávať pozadie. Hory, kopce, lúky či mesto. Podľa vašej fantázie, v akom prostredí sa má dej odohrávať, vyplňte text, a vyberte na akú stranu a do akého okienka chcete toto pozadie nakresliť.

Pozor, opäť pridávať môžete iba do strán či okienok, ktoré ste už vytvorili!

Kliknutím na tlačidlo **Namaluj pozadie**, pozadie uložíte.

**Maliari postáv:**

Najskôr si najmete prvého maliara tlačidlom **Vytvoriť nového maliara postavy.**

Môžete ich vytvoriť koľko sa vám zapáči. Každý maliar ale kreslí len svoju postavičku, takže mu ju musíte najprv priradiť. Napr. do prvého okna zadáte číslo 1 (prvý maliar), a do druhého textového okna *napr. meno "Guliver"* a vyberiete že má byť bojovník. Tlačidlom **Ulož postavu pre maliara** postavu uložíte. Potom vytvoríte druhého maliara, a zadáte číslo 2, *a meno postavy napr "Lucia"* a vyberiete že má byť teenager a znova uložíte.

Odomklo sa vám tlačidlo **Malovanie.**

Okienka znova vyplníte, napr. maliar číslo 1 (kreslí Gulivera) na stranu číslo 1 do okienka číslo 1 na miesto postavy číslo 1. Môžete zaškrknúť políčko "Ozbrojená?", ak chcete aby mala na danej snímke svoju zbraň. Tlačidlom **Namaluj postavu** ju uložíte. Podobne viete vedľa tejto postavy druhým maliarom dokresliť Luciu.

Pozor, opäť pridávať môžete iba do strán, okienok či na miesta postáv, ktoré ste už vytvorili!

V textovom poli na predchádzajúcom okne vidíte, čo ste vytvorili, a aj zbraň. Guliver má v tomto prípade Katanu, čo je v japonsku najpoužívanejšia zbraň bojovníkov či ninjov.

**Výpisy:**

A posledným tlačidlom v **Hlavnom Okne** je tlačidlo **Výpisy.**

V tomto okne viete zistiť pár základných informácii o vašom komixe: názov, počet strán, a aké všetky postavy vedia maliari kresliť.

*Uvedený príklad dohromady vytvoril komix Guliverove cesty s jednou stranou a jedným okienkom, v ktorom sa nachádzali 2 postavy: Guliver a Lucia, priatelia, ozbrojený bojovník (možno ninja) a teenagerka, ktorí spolu vedú rozhovor.*