IceAge

1.0

Generováno programem Doxygen 1.8.11

Contents

Chapter 1

Rejstřík prostorů jmen

1.1 Seznam prostorů jmen

Zde naleznete seznam všech prostorů jmen se stručným popisem:

Chapter 2

Rejstřík hierarchie tříd

2.1 Hierarchie tříd

Zde naleznete seznam, vyjadřující vztah dědičnosti tříd. Je seřazen přibližně (ale ne úplně) podle abecedy:

Chapter 3

Rejstřík tříd

3.1 Seznam tříd

Následující seznam obsahuje především identifikace tříd, ale nacházejí se zde i další netriviální prvky, jako jsou struktury (struct), unie (union) a rozhraní (interface). V seznamu jsou uvedeny jejich stručné popisy:

Chapter 4

Rejstřík souborů

4.1 Seznam souborů

Zde naleznete seznam všech souborů se stručnými popisy:

Chapter 5

Dokumentace prostorů jmen

5.1 Dokumentace prostoru jmen IceAge

Třídy

- class Beast
- class BuilderBytost
- class BuilderLokace
- class BuilderPredmet
- class Command
- class EngineIceAge
- class Hra
- class Humanoid
- class IInventar
- class IMemory
- class INazev
- class Inventar
- class IOblecenéVybaveni
- class IPopis
- class IStaty
- class IZivot
- class Jezero
- class JizniLes
- class Ledovec
- class LocationDirector
- class Lokace
- class Memory
- class MemoryArbiter
- class Nazev
- class ObleceneVybaveniHumanoid
- class ObleceneVybaveniVeverky
- class Orisek
- class Popis
- class Predmet
- class Reka
- class SeverniLes
- class Staty
- class Veverka

- class VychodniBrehReky
- class VychodniPlanina
- class ZapadniBrehReky
- class ZapadniPlanina
- class Zbran
- class Zivot

Funkce

- void boj (IceAge::Zivot *utocnik, IceAge::Zivot *obrance)

5.1.1 Dokumentace funkcí

5.1.1.1 void IceAge::boj (IceAge::Zivot * *utocnik*, IceAge::Zivot * *obrance*)

Funkce, ktera zprostredkovava boj mezi dvema potomky tridy Zivot.

Funkce, ktera zprostredkovava boj mezi dvema potomky tridy Zivot. Boj probiha jednokolove. Nejprve utoci utocnik, nasledne utoci obrance. Pokud nekdo z nich zemre, hra se ukonci. O boji se vypisuje informace.

Definice je uvedena na řádku 12 v souboru VolneFunkce.cpp.

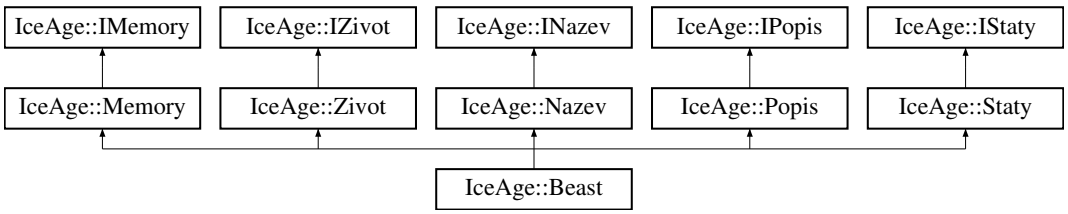
Chapter 6

Dokumentace tříd

6.1 Dokumentace třídy IceAge::Beast

`#include <Beast.h>`

Diagram dědičnosti pro třídu IceAge::Beast



Veřejné metody

- short utok ()

  Konkretni implementace dedene metody vracejici hodnotu celkoveho utoku.
- short getZurivost ()

  Metoda vracejici hodnotu zurivosti.
- short obrana ()

  Konkretni implementace dedene metody vracejici hodnotu celkove obrany.
- Beast (short novaZurivost, std::string novyNazev, short novaSila, std::string novyPopis, short novaOdolnost, short novyZivot)

  Konstruktor.
- ~Beast ()

  Destruktor.

Privátní metody

- void setZurivost (short novaZurivost)

  Metoda nastavujici zurivost.

Privátní atributy

- short m_zurivost

  Atribut znazornujici jak agresivni muze dane zvire byt. Muze byt i zaporna.

Další zděděné členy

6.1.1 Detailní popis

Jedna se o tridu, ktera znazornuje zvirata, ktera nemohou nest zadne veci. Ridi se spise instinkty.

Definice je uvedena na řádku 17 v souboru Beast.h.

6.1.2 Dokumentace konstruktoru a destruktoru

6.1.2.1 IceAge::Beast::Beast (short *novaZurivost,* std::string *novyNazev,* short *novaSila,* std::string *novyPopis,* short *novaOdolnost,* short *novyZivot*)

Konstruktor.

Konstruktor, ve kterem se nastavuje zurivost

Definice je uvedena na řádku 37 v souboru Beast.cpp.

6.1.2.2 IceAge::Beast::~Beast ( )

Destruktor.

Destruktor

Definice je uvedena na řádku 45 v souboru Beast.cpp.

6.1.3 Dokumentace k metodám

6.1.3.1 short IceAge::Beast::getZurivost ( )

Metoda vracejici hodnotu zurivosti.

Metoda vracejici hodnotu zurivosti.

Definice je uvedena na řádku 9 v souboru Beast.cpp.

6.1.3.2 short IceAge::Beast::obrana ( ) [virtual]

Konkretni implementace dedene metody vracejici hodnotu celkove obrany.

Konkretni implementace dedene metody vracejici hodnotu celkove obrany.

Implementuje IceAge::Zivot.

Definice je uvedena na řádku 30 v souboru Beast.cpp.

6.1.3.3 void IceAge::Beast::setZurivost (short *novaZurivost*) [private]

Metoda nastavujici zurivost.

Metoda nastavujici hodnotu zurivosti.

Definice je uvedena na řádku 16 v souboru Beast.cpp.

6.1.3.4 short IceAge::Beast::utok ( ) [virtual]

Konkretni implementace dedene metody vracejici hodnotu celkoveho utoku.

Konkretni implementace dedene metody vracejici hodnotu celkoveho utoku.

Implementuje IceAge::Zivot.

Definice je uvedena na řádku 23 v souboru Beast.cpp.

6.1.4 Dokumentace k datovým členům

6.1.4.1 short IceAge::Beast::m_zurivost [private]

Atribut znazornujici jak agresivni muze dane zvire byt. Muze byt i zaporna.

Definice je uvedena na řádku 20 v souboru Beast.h.

Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/Beast.h
- cpp/Beast.cpp

6.2 Dokumentace třídy IceAge::BuilderBytost

`#include <BuilderBytost.h>`

Veřejné metody

- BuilderBytost (IceAge::BuilderPredmet *novyBuilder)
  Konstruktor.
- ~BuilderBytost ()
  Destruktor.
- void setBuilder (IceAge::BuilderPredmet *novyBuilder)
  Metoda ktera nastavuje BuilderPredmet.
- IceAge::Predmet * createPredmet (std::string novyNazev, short novaHodnota)
  Metoda vytvarejici predmet.
- IceAge::Beast * getBeast ()
  Metoda vracejici Beast.
- IceAge::Humanoid * getHumanoid ()
  Metoda vracejici Humanoida.
- void createBeast (short novaZurivost, std::string novyNazev, short novaSila, std::string novyPopis, short novaOdolnost, short novyZivot)
  Metoda vytvarejici Beast.
- void createHumanoid (short novyZivot, std::string novyPopis, short novaSila, short novaOdolnost, std::string novyNazevPredmetu1, std::string novyNazevPredmetu2, std::string novyNazevPredmetu3, short nova↩ HodnotaPredmetu1, short novaHodnotaPredmetu2, short novaHodnotaPredmetu3)
  Metoda vytvarejici Humanoida.

Privátní atributy

- IceAge::Beast * m_beast

  Atribut do ktereho se uklada odkaz vytvareneho beastu.
- IceAge::Humanoid * m_humanoid

  Atribut do ktereho se uklada odkaz vytvareneho humanoida.
- IceAge::BuilderPredmet * m_builder

  Atribut do ktereho se uklada odkaz na BuilderPredmet.

6.2.1 Detailní popis

Trida, ktera bude stavet bytosti(Humanoidy s predmety/Beasty). Podle navrhoveho vzoru Builder.

Definice je uvedena na řádku 16 v souboru BuilderBytost.h.

6.2.2 Dokumentace konstruktoru a destruktoru

6.2.2.1 IceAge::BuilderBytost::BuilderBytost (IceAge::BuilderPredmet * *novyBuilder*)

Konstruktor.

Konstruktor ve kterem se nastavuje novy BuilderPredmet a do ostatnich atributu se uklada nullptr.

Definice je uvedena na řádku 9 v souboru BuilderBytost.cpp.

6.2.2.2 IceAge::BuilderBytost::~BuilderBytost ( )

Destruktor.

Metoda vytvarejici Humanoida.

Definice je uvedena na řádku 69 v souboru BuilderBytost.cpp.

6.2.3 Dokumentace k metodám

6.2.3.1 void IceAge::BuilderBytost::createBeast (short *novaZurivost*, std::string *novyNazev*, short *novaSila*, std::string *novyPopis*, short *novaOdolnost*, short *novyZivot*)

Metoda vytvarejici Beast.

Metoda vytvarejici Beast.

Definice je uvedena na řádku 52 v souboru BuilderBytost.cpp.

6.2.3.2 void IceAge::BuilderBytost::createHumanoid (short *novyZivot,* std::string *novyPopis,* short *novaSila,* short *novaOdolnost,* std::string *novyNazevPredmetu1,* std::string *novyNazevPredmetu2,* std::string *novyNazevPredmetu3,* short *novaHodnotaPredmetu1,* short *novaHodnotaPredmetu2,* short *novaHodnotaPredmetu3*)

Metoda vytvarejici Humanoida.

Definice je uvedena na řádku 56 v souboru BuilderBytost.cpp.

6.2.3.3 IceAge::Predmet * IceAge::BuilderBytost::createPredmet (std::string *novyNazev,* short *novaHodnota*)

Metoda vytvarejici predmet.

Metoda vytvarejici predmet. Nejdrive zavola metodu pro vytvoreni buildera, nasledne zavola getter.

Definice je uvedena na řádku 26 v souboru BuilderBytost.cpp.

6.2.3.4 IceAge::Beast * IceAge::BuilderBytost::getBeast ( )

Metoda vracejici Beast.

Metoda vracejici Beast. Nastavuje atribut beast na nullptr.

Definice je uvedena na řádku 34 v souboru BuilderBytost.cpp.

6.2.3.5 IceAge::Humanoid * IceAge::BuilderBytost::getHumanoid ( )

Metoda vracejici Humanoida.

Metoda vracejici Humanoida. Nastavuje atribut humanoid na nullptr.

Definice je uvedena na řádku 43 v souboru BuilderBytost.cpp.

6.2.3.6 void IceAge::BuilderBytost::setBuilder (IceAge::BuilderPredmet * *novyBuilder*)

Metoda ktera nastavuje BuilderPredmet.

Metoda ktera nastavuje BuilderPredmet.

Definice je uvedena na řádku 19 v souboru BuilderBytost.cpp.

6.2.4 Dokumentace k datovým členům

6.2.4.1 IceAge::Beast* IceAge::BuilderBytost::m_beast [private]

Atribut do ktereho se uklada odkaz vytvareneho beastu.

Definice je uvedena na řádku 19 v souboru BuilderBytost.h.

6.2.4.2 IceAge::BuilderPredmet* IceAge::BuilderBytost::m_builder [private]

Atribut do ktereho se uklada odkaz na BuilderPredmet.

Definice je uvedena na řádku 21 v souboru BuilderBytost.h.

6.2.4.3 IceAge::Humanoid* IceAge::BuilderBytost::m_humanoid [private]

Atribut do ktereho se uklada odkaz vytvareneho humanoida.

Definice je uvedena na řádku 20 v souboru BuilderBytost.h.

Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/BuilderBytost.h
- cpp/BuilderBytost.cpp

6.3 Dokumentace třídy IceAge::BuilderLokace

`#include <BuilderLokace.h>`

Veřejné metody

- BuilderLokace (IceAge::BuilderBytost *novyBuilder)

  Konstruktor.
- ~BuilderLokace ()

  Destruktor.
- void setBuilder (IceAge::BuilderBytost *novyBuilder)

  Metoda ktera nastavuje BuilderBytost.
- void createLokace ()

  Metoda vytvarejici Lokaci.
- IceAge::Lokace * getLokace ()

  Metoda vracejici Lokaci.
- IceAge::Humanoid * createHumanoid (short novyZivot, std::string novyPopis, short novaSila, short novaOdolnost, std::string novyNazevPredmetu1, std::string novyNazevPredmetu2, std::string novyNazevPredmetu3, short novaHodnotaPredmetu1, short novaHodnotaPredmetu2, short novaHodnotaPredmetu3)

  Metoda vytvarejici humanoida.
- IceAge::Beast * createBeast (short novaZurivost, std::string novyNazev, short novaSila, std::string novyPopis, short novaOdolnost, short novyZivot)

  Metoda vytvarejici beast.
- void nactiBeasty (std::ifstream &popisLokace)

  Metoda ktera nacte beasty ze souboru a ulozi je do lokace.
- void nactiHumanoidy (std::ifstream &popisLokace)

  Metoda ktera nacte humanoidy ze souboru a ulozi je do lokace.

Privátní atributy

- IceAge::BuilderBytost * m_builderBytost

  Atribut obsahujici odkaz na BuilderBytost.
- IceAge::Lokace * m_lokace

  Atribut obsahujici odkaz na prave vytvarenou Lokaci.
- std::vector< IceAge::Lokace * > m_vektorLokaci

  Vector jiz hotovych lokaci, ktere se nasledne spoji do jedne "mapy".

6.3.1 Detailní popis

Trida, ktera bude stavet Lokace (ktera obsahuje beasty/humanoidy). Podle navrhoveho vzoru Builder.

Definice je uvedena na řádku 18 v souboru BuilderLokace.h.

6.3.2 Dokumentace konstruktoru a destruktoru

6.3.2.1 IceAge::BuilderLokace::BuilderLokace (IceAge::BuilderBytost * *novyBuilder*)

Konstruktor.

Konstruktor ve kterem se nastavuje BuilderBytost a do lokace se uklada nullptr.

Definice je uvedena na řádku 9 v souboru BuilderLokace.cpp.

6.3.2.2 IceAge::BuilderLokace::~BuilderLokace ( )

Destruktor.

Destruktor ve kterem se maze builder a pripadne vytvarena lokace.

Definice je uvedena na řádku 207 v souboru BuilderLokace.cpp.

6.3.3 Dokumentace k metodám

6.3.3.1 IceAge::Beast * IceAge::BuilderLokace::createBeast (short *novaZurivost*, std::string *novyNazev*, short *novaSila*, std::string *novyPopis*, short *novaOdolnost*, short *novyZivot*)

Metoda vytvarejici beast.

Metoda vytvarejici beasta. Provola metody ostatnich builderu a vrati vytvoreneho beasta.

Definice je uvedena na řádku 42 v souboru BuilderLokace.cpp.

6.3.3.2 IceAge::Humanoid * IceAge::BuilderLokace::createHumanoid (short *novyZivot,* std::string *novyPopis,* short *novaSila,* short *novaOdolnost,* std::string *novyNazevPredmetu1,* std::string *novyNazevPredmetu2,* std::string *novyNazevPredmetu3,* short *novaHodnotaPredmetu1,* short *novaHodnotaPredmetu2,* short *novaHodnotaPredmetu3*)

Metoda vytvarejici humanoida.

Metoda vytvarejici humanoida. Provola metody ostatnich builderu a vrati vytvoreneho Humanoida.

Definice je uvedena na řádku 31 v souboru BuilderLokace.cpp.

6.3.3.3 void IceAge::BuilderLokace::createLokace ( )

Metoda vytvarejici Lokaci.

Metoda vytvarejici Lokaci. <Otevreni seznamu lokaci.

<V pripade chyby uvolni pamet a skonci.

<Lokalni promenne, do kterych se bude nacitat ze souboru.

<Nacitani nazvu souboru ze souboru ".seznam"

<Otevirani souboru s popisem aktualne vytvarene lokace.

<V pripade chyby uvolni pamet a skonci.

Nastavovani zakladnich atributu lokace.

<Vytvoreni hole kostry lokace.

<Rozhodnuti, zda se vytvari reka

<Nacteni humanoidu ze souboru.

<Nacterni beastu ze souboru.

<Cyklus ve kterem se usporadaji odkazy mezi jednotlivymi lokacemi.

<Do lokace se vlozi odpovidajici trida s metodou pruzkumu.

<Ulozeni lokace do vectoru.

<Uzavreni souboru s popisem aktualni lokace.

<Uzavreni souboru se seznamem souboru vsech lokaci.

<Nastaveni prvni lokace jako hlavni (aktualni).

Definice je uvedena na řádku 50 v souboru BuilderLokace.cpp.

6.3.3.4 IceAge::Lokace * IceAge::BuilderLokace::getLokace ( )

Metoda vracejici Lokaci.

Metoda vracejici Lokaci.

Definice je uvedena na řádku 24 v souboru BuilderLokace.cpp.

6.3.3.5 void IceAge::BuilderLokace::nactiBeasty (std::ifstream & *popisLokace*)

Metoda ktera nacte beasty ze souboru a ulozi je do lokace.

Metoda vytvarejici Beasty na zaklade popisu v souboru. Pokud jsou spatne popsane lokace v souborech, nactou se spatne a muze dojit ke spatnemu pristupu do pameti. <Lokalni promena znacici pocet Beastu v lokaci.

<Nacteni poctu Beastu + ignorace konce radku.

<Pokud beasty v lokaci nejsou, nacte znacku a ukonci metodu.

<Lokalni promenne potrebne k vytvoreni beasta.

<Cyklus ve kterem se nactou hodnoty potrebne pro vytvoreni beasta a nasledne jeho vytvoreni a pridani do vectoru.

<Odebrani znacky.

Definice je uvedena na řádku 173 v souboru BuilderLokace.cpp.

6.3.3.6 void IceAge::BuilderLokace::nactiHumanoidy (std::ifstream & *popisLokace*)

Metoda ktera nacte humanoidy ze souboru a ulozi je do lokace.

Metoda vytvarejici Humanoidy na zaklade popisu v souboru. Pokud jsou spatne popsane lokace v souborech, nactou se spatne a muze dojit ke spatnemu pristupu do pameti. <Lokalni promena znacici pocet Humanoidu v lokaci.

<Nacteni poctu Humanoidu + ignorace konce radku.

<Pokud humanoidi v lokaci nejsou, nacte znacku a ukonci metodu.

<Lokalni promenne potrebne k vytvoreni humanoida.

<Cyklus ve kterem se nactou hodnoty potrebne pro vytvoreni humanoida a nasledne jeho vytvoreni a pridani do vectoru.

<Odebrani znacky.

Definice je uvedena na řádku 132 v souboru BuilderLokace.cpp.

6.3.3.7 void IceAge::BuilderLokace::setBuilder (IceAge::BuilderBytost * *novyBuilder*)

Metoda ktera nastavuje BuilderBytost.

Metoda ktera nastavuje BuilderBytost.

Definice je uvedena na řádku 17 v souboru BuilderLokace.cpp.

6.3.4 Dokumentace k datovým členům

6.3.4.1 IceAge::BuilderBytost* IceAge::BuilderLokace::m_builderBytost [private]

Atribut obsahujici odkaz na BuilderBytost.

Definice je uvedena na řádku 21 v souboru BuilderLokace.h.

6.3.4.2 IceAge::Lokace* IceAge::BuilderLokace::m_lokace [private]

Atribut obsahujici odkaz na prave vytvarenou Lokaci.

Definice je uvedena na řádku 22 v souboru BuilderLokace.h.

6.3.4.3 std::vector<IceAge::Lokace*> IceAge::BuilderLokace::m_vektorLokaci [private]

Vector jiz hotovych lokaci, ktere se nasledne spoji do jedne "mapy".

Definice je uvedena na řádku 23 v souboru BuilderLokace.h.

Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/BuilderLokace.h
- cpp/BuilderLokace.cpp

6.4 Dokumentace třídy IceAge::BuilderPredmet

`#include <BuilderPredmet.h>`

Veřejné metody

- void createPredmet (std::string novyNazev, short novaHodnota)

  Metoda, ktera vytvori predmet.
- IceAge::Predmet * getPredmet ()

  Metoda vracejici odkaz na vytvareny predmet.
- BuilderPredmet ()

  Konstruktor.
- ~BuilderPredmet ()

  Destruktor.

Privátní atributy

- IceAge::Predmet * m_pripravovany

  Atribut do ktereho se uklada odkaz vytvareneho predmetu.

6.4.1 Detailní popis

Trida, ktera bude stavet predmety. Podle navrhoveho vzoru Builder.

Definice je uvedena na řádku 14 v souboru BuilderPredmet.h.

6.4.2 Dokumentace konstruktoru a destruktoru

6.4.2.1 IceAge::BuilderPredmet::BuilderPredmet ( )

Konstruktor.

Konstruktor ve kterem se jen do atributu pripravovany ulozi nullptr;

Definice je uvedena na řádku 25 v souboru BuilderPredmet.cpp.

6.4.2.2 IceAge::BuilderPredmet::∼BuilderPredmet ( )

Destruktor.

Destruktor ve kterem se pripadne maze vytvareny predmet.

Definice je uvedena na řádku 32 v souboru BuilderPredmet.cpp.

6.4.3 Dokumentace k metodám

6.4.3.1 void IceAge::BuilderPredmet::createPredmet (std::string *novyNazev,* short *novaHodnota*)

Metoda, ktera vytvori predmet.

Metoda, ktera vytvori predmet podle zadanych parametru.

Definice je uvedena na řádku 9 v souboru BuilderPredmet.cpp.

6.4.3.2 IceAge::Predmet ∗ IceAge::BuilderPredmet::getPredmet ( )

Metoda vracejici odkaz na vytvareny predmet.

Metoda ktera vraci odkaz na predmet, ktery je vytvaren. Nasledne do atributu pripravovany ulozi nullptr.

Definice je uvedena na řádku 16 v souboru BuilderPredmet.cpp.

6.4.4 Dokumentace k datovým členům

6.4.4.1 IceAge::Predmet∗ IceAge::BuilderPredmet::m_pripravovany [private]

Atribut do ktereho se uklada odkaz vytvareneho predmetu.

Definice je uvedena na řádku 17 v souboru BuilderPredmet.h.

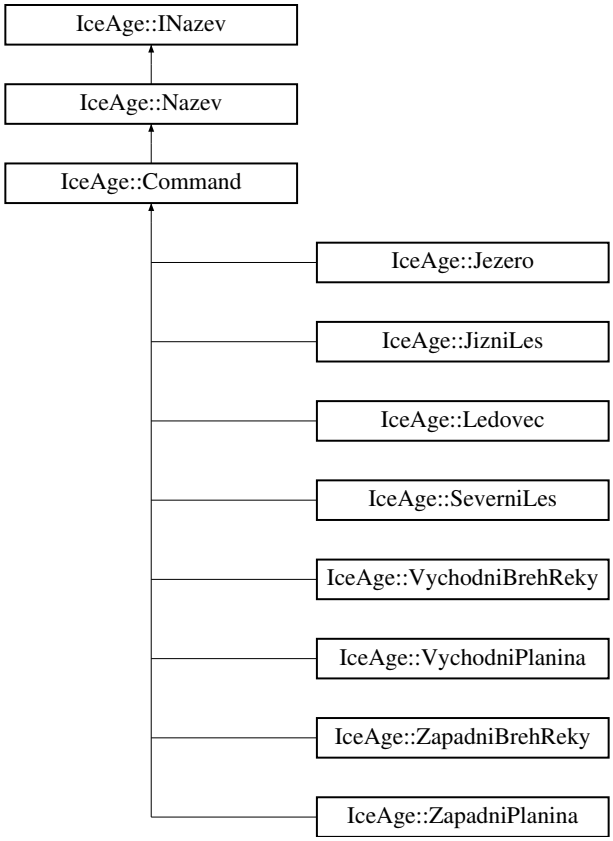
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/BuilderPredmet.h
- cpp/BuilderPredmet.cpp

6.5 Dokumentace třídy IceAge::Command

`#include <Command.h>`

Diagram dědičnosti pro třídu IceAge::Command



Veřejné metody

- Command (std::string novyNazev)

  Konstruktor.
- virtual ~Command ()

  Virtualni Destruktor.
- virtual void prozkoumat (IceAge::Veverka *scrat, std::vector< IceAge::Beast * > vektorBeast, std::vector< IceAge::Humanoid * > vektorHumanoid, IceAge::Reka *reka)=0

  Virtualni metoda, pomoci ktere se prida nova metoda(funkcionalita/algoritmus) do jiz existujici tridy.

Další zděděné členy

6.5.1 Detailní popis

Abstraktni trida vyuzivana pro navrhovy vzor Command. Jednotlivi potomci jsou sparovani s jednotlivymi instancemi lokace v pomeru 1:1. Diky tomu pujde prozkoumavat tyto lokace.

Definice je uvedena na řádku 20 v souboru Command.h.

6.5.2 Dokumentace konstruktoru a destruktoru

6.5.2.1 IceAge::Command::Command (std::string *novyNazev*)

Konstruktor.

Konstruktor

Definice je uvedena na řádku 9 v souboru Command.cpp.

6.5.2.2 IceAge::Command::~Command ( ) [virtual]

Virtualni Destruktor.

Destruktor

Definice je uvedena na řádku 15 v souboru Command.cpp.

6.5.3 Dokumentace k metodám

6.5.3.1 virtual void IceAge::Command::prozkoumat (IceAge::Veverka * *scrat*, std::vector< IceAge::Beast * > *vektorBeast*, std::vector< IceAge::Humanoid * > *vektorHumanoid*, IceAge::Reka * *reka*) [pure virtual]

Virtualni metoda, pomoci ktere se prida nova metoda(funkcionalita/algoritmus) do jiz existujici tridy.

Implementováno v IceAge::Jezero, IceAge::Ledovec, IceAge::ZapadniBrehReky, IceAge::JizniLes, IceAge::SeverniLes, IceAge::VychodniBrehReky, IceAge::VychodniPlanina a IceAge::ZapadniPlanina.

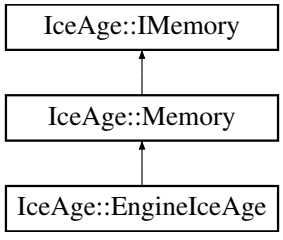
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/Command.h
- cpp/Command.cpp

6.6 Dokumentace třídy IceAge::EngineIceAge

```
#include <EngineIceAge.h>
```

Diagram dědičnosti pro třídu IceAge::EngineIceAge



Veřejné metody

- void startHry ()

  Metoda spoustejici hru.
- ~EngineIceAge ()

  Destruktor.

Statické veřejné metody

- static IceAge::EngineIceAge * getInstance ()

  Metoda vracejici instanci teto tridy.

Privátní metody

- EngineIceAge (IceAge::LocationDirector *novyDirector)

  Konstruktor.
- IceAge::Veverka * getVeverka ()

  Metoda vracejici Veverku.
- void createVeverka ()

  Metoda vytvarejici Veverku.
- bool inicializaceLokaci ()

  Metoda inicializujici Lokace.
- void setDirector (IceAge::LocationDirector *novyDirector)

  Metoda nastavujici noveho LocationDirector.

Privátní atributy

- IceAge::Veverka * m_veverka

  Atribut udrzujici odkaz na Veverku.
- IceAge::LocationDirector * m_directorLokaci

  Atribut udrzujici odkaz na LocationDirector.
- IceAge::Lokace * m_lokace

  Atribut udrzujici odkaz na aktualni lokaci.

Statické privátní atributy

- static IceAge::EngineIceAge * s_vlastniInstance =nullptr

  Staticky atribut udrzujici odkaz sam na sebe.

Další zděděné členy

6.6.1 Detailní popis

Trida ktera zastituje celou hru. Inicializuje vsechny soubory a vetsinu trid. Spousti hru. Trida je implementovana jako Singleton.

Definice je uvedena na řádku 17 v souboru EngineIceAge.h.

6.6.2 Dokumentace konstruktoru a destruktoru

6.6.2.1 IceAge::EngineIceAge::EngineIceAge (IceAge::LocationDirector * *novyDirector*) [private]

Konstruktor.

Konstruktor ve kterem se nastavuje LocationDirector a zbytek atributu se nastavuje na nullptr.

Definice je uvedena na řádku 22 v souboru EngineIceAge.cpp.

6.6.2.2 IceAge::EngineIceAge::~EngineIceAge ( )

Destruktor.

Destruktor ve kterem se mazou odkazy z atributu.

Definice je uvedena na řádku 78 v souboru EngineIceAge.cpp.

6.6.3 Dokumentace k metodám

6.6.3.1 void IceAge::EngineIceAge::createVeverka ( ) [private]

Metoda vytvarejici Veverku.

Metoda vytvarejici Veverku.

Definice je uvedena na řádku 38 v souboru EngineIceAge.cpp.

6.6.3.2 IceAge::EngineIceAge * IceAge::EngineIceAge::getInstance ( ) [static]

Metoda vracejici instanci teto tridy.

Metoda vracejici instanci teto tridy. Pokud jiz existuje, vrati odkaz na ni.

Definice je uvedena na řádku 11 v souboru EngineIceAge.cpp.

6.6.3.3 IceAge::Veverka * IceAge::EngineIceAge::getVeverka ( ) [private]

Metoda vracejici Veverku.

Metoda vracejici Veverku.

Definice je uvedena na řádku 31 v souboru EngineIceAge.cpp.

6.6.3.4 bool IceAge::EngineIceAge::inicializaceLokaci ( ) [private]

Metoda inicializujici Lokace.

Metoda inicializujici Lokace. Provola se LocationDirector a v pripade uspechu se vrati true.

Definice je uvedena na řádku 62 v souboru EngineIceAge.cpp.

6.6.3.5 void IceAge::EngineIceAge::setDirector (IceAge::LocationDirector * *novyDirector*) [private]

Metoda nastavujici noveho LocationDirector.

Metoda nastavujici noveho LocationDirector.

Definice je uvedena na řádku 71 v souboru EngineIceAge.cpp.

6.6.3.6 void IceAge::EngineIceAge::startHry ( )

Metoda spoustejici hru.

Metoda spoustejici hru. Pokud se provede spravne inicializace lokaci a vytvori se veverka, vytvori tridu IceAge::Hra a spusti hru.

Definice je uvedena na řádku 45 v souboru EngineIceAge.cpp.

6.6.4 Dokumentace k datovým členům

6.6.4.1 IceAge::LocationDirector* IceAge::EngineIceAge::m_directorLokaci [private]

Atribut udrzujici odkaz na LocationDirector.

Definice je uvedena na řádku 22 v souboru EngineIceAge.h.

6.6.4.2 IceAge::Lokace* IceAge::EngineIceAge::m_lokace [private]

Atribut udrzujici odkaz na aktualni lokaci.

Definice je uvedena na řádku 23 v souboru EngineIceAge.h.

6.6.4.3 IceAge::Veverka* IceAge::EngineIceAge::m_veverka [private]

Atribut udrzujici odkaz na Veverku.

Definice je uvedena na řádku 20 v souboru EngineIceAge.h.

6.6.4.4 IceAge::EngineIceAge * IceAge::EngineIceAge::s_vlastniInstance =nullptr [static], [private]

Staticky atribut udrzujici odkaz sam na sebe.

Inicializace statickeho atributu.

Definice je uvedena na řádku 21 v souboru EngineIceAge.h.

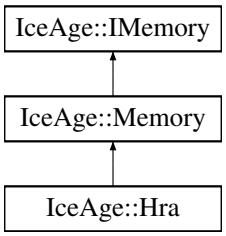
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/EngineIceAge.h
- cpp/EngineIceAge.cpp

6.7 Dokumentace třídy IceAge::Hra

`#include <Hra.h>`

Diagram dědičnosti pro třídu IceAge::Hra



Veřejné metody

- Hra (IceAge::Lokace *novaAktualniLokace)
  Konstruktor.
- void zacniHrat (IceAge::Veverka *scrat)
  Metoda ktera spusti pribeh.

Privátní metody

- void zmenLokaci (IceAge::Veverka *scrat)
  Metoda ktera zmeni aktualni lokaci.
- void zmenLokaci (unsigned int index, IceAge::Veverka *scrat)
  Metoda ktera zmeni aktualni lokaci.
- void setAktualniLokace (IceAge::Lokace *novaAktualniLokace)
  Metoda ktera nastavi aktualni lokaci.
- void vypisAktualniLokaci ()
  Metoda ktera vypise aktualni lokaci.
- void vypisAsciObrazek ()
  Metoda ktera vypise aktualni obrazek.
- void intro (IceAge::Veverka *scrat)
  Metoda ktera je uvodnim algoritmem(dejem) ve hre.

Privátní atributy

- IceAge::Lokace * m_aktualniLokace
  Atribut udrzujici odkaz na aktualni lokaci.
- short m_obrazek
  Atribut ktery je pocitadlo aktualniho obrazku.

Další zděděné členy

6.7.1 Detailní popis

Trida zajistujici samotny prubeh hry.

Definice je uvedena na řádku 15 v souboru Hra.h.

6.7.2 Dokumentace konstruktoru a destruktoru

6.7.2.1 IceAge::Hra::Hra (IceAge::Lokace * *novaAktualniLokace*)

Konstruktor.

Konstruktor ktery nastavi aktualni lokaci a aktualni obrazek na 0;

Definice je uvedena na řádku 9 v souboru Hra.cpp.

6.7.3 Dokumentace k metodám

6.7.3.1 void IceAge::Hra::intro (IceAge::Veverka * *scrat*) [private]

Metoda ktera je uvodnim algoritmem(dejem) ve hre.

<Lokalni promenna pozivana pro nacitani odpovedi od uzivatele.

<Lokalni promenne pouzivane pro cykleni.

<Lokalni promenna pouzivana pro cislovani moznosti, ktere se zobrazi uzivateli.

<Vypise obrazek na obrazovku.

<Vypise obrazek na obrazovku.

<Cykleni pri cekani na spravne hodnoty od hrace.

<Reset lokalni promenne.

<Cykleni vypisu s moznostmi akci.

<Cykleni pri cekani na spravne hodnoty od hrace.

<Moznost Hledej dal

<Moznost Vloz orisek

<Moznost Kde ze jsem?

<Reset lokalnich promennych

<Vypis obrazek

<Zmena lokace

<Cykleni pri cekani na spravne hodnoty od hrace. Spravna je pouze jedna hodnota.

<Reset lokalni promenne

Interakce se Supomutem.

<Cykleni pri cekani na spravnou hodnotu od hrace

<Moznost Zautocit

<Moznost Utect

<Moznost Schovat se

<Reset lokalni promenne

<Cykleni pri cekani na spravnou jednu hodnotu

<Zmena lokace na finalni v intru

Definice je uvedena na řádku 194 v souboru Hra.cpp.

6.7.3.2 void IceAge::Hra::setAktualniLokace (IceAge::Lokace * *novaAktualniLokace*) [private]

Metoda ktera nastavi aktualni lokaci.

Metoda ktera nastavi aktualni lokaci.

Definice je uvedena na řádku 66 v souboru Hra.cpp.

6.7.3.3 void IceAge::Hra::vypisAktualniLokaci ( ) [private]

Metoda ktera vypise aktualni lokaci.

Metoda ktera vypise aktualni lokaci, tj jeho nazev a popis.

Definice je uvedena na řádku 73 v souboru Hra.cpp.

6.7.3.4 void IceAge::Hra::vypisAsciObrazek ( ) [private]

Metoda ktera vypise aktualni obrazek.

Metoda ktera vypise aktualni obrazek. Rozhoduje se na zaklade atributu obrazek.

Definice je uvedena na řádku 83 v souboru Hra.cpp.

6.7.3.5 void IceAge::Hra::zacniHrat (IceAge::Veverka * *scrat*)

Metoda ktera spusti pribeh.

Metoda ktera spusti pribeh. <Lokalni promenna pozivana pro nacitani odpovedi od uzivatele.

<Lokalni promenne pouzite pro cykly.

<Lokalni promenna pouzivana pro cislovani moznosti, ktere se zobrazi uzivateli.

<Spust uvodni dej.

<Cykleni v lokacich

<Cykleni v ziskavani odpovedi od hrace

<Moznost Jit jinam

<Moznost Kde ze jsem?

<Prozkoumat to tu

Definice je uvedena na řádku 336 v souboru Hra.cpp.

6.7.3.6 void IceAge::Hra::zmenLokaci (IceAge::Veverka * *scrat*) [private]

Metoda ktera zmeni aktualni lokaci.

Metoda ktera zmeni lokaci na zaklade hracova vyberu.

Definice je uvedena na řádku 17 v souboru Hra.cpp.

6.7.3.7 void IceAge::Hra::zmenLokaci (unsigned int *index,* IceAge::Veverka * *scrat*) [private]

Metoda ktera zmeni aktualni lokaci.

Metoda ktera zmeni lokaci na zaklade hodnoty indexu a pripadnemu vybaveni hrace. <Pokud se cestuje z lesa do lesa.

<Pokud nevlastnis lod a klacek

<Pokud se cestuje z lesa do lesa.

Definice je uvedena na řádku 40 v souboru Hra.cpp.

6.7.4 Dokumentace k datovým členům

6.7.4.1 IceAge::Lokace* IceAge::Hra::m_aktualniLokace [private]

Atribut udrzujici odkaz na aktualni lokaci.

Definice je uvedena na řádku 18 v souboru Hra.h.

6.7.4.2 short IceAge::Hra::m_obrazek [private]

Atribut ktery je pocitadlo aktualniho obrazku.

Definice je uvedena na řádku 19 v souboru Hra.h.

Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/Hra.h
- cpp/Hra.cpp

6.8 Dokumentace třídy IceAge::Humanoid

`#include <Humanoid.h>`

Diagram dědičnosti pro třídu IceAge::Humanoid

Veřejné metody

- Humanoid (short novyZivot, std::string novyNazev, std::string novyPopis, short novaSila, short novaOdolnost, IceAge::Predmet ∗zbran1, IceAge::Predmet ∗zbran2, IceAge::Predmet ∗zbran3)

  Konstruktor.
- ∼Humanoid ()

  Destruktor.
- short utok ()

  Podedena metoda vracejici hodnotu celkoveho utoku.
- short obrana ()

  Podedena metoda vracejici hodnotu celkove obrany.

Další zděděné členy

6.8.1 Detailní popis

Jedna se o tridu, ktera znazornuje Lidi, kteri mohou nest veci i na zadech.

Definice je uvedena na řádku 19 v souboru Humanoid.h.

6.8.2 Dokumentace konstruktoru a destruktoru

6.8.2.1 IceAge::Humanoid::Humanoid (short *novyZivot*, std::string *novyNazev*, std::string *novyPopis*, short *novaSila*, short *novaOdolnost*, IceAge::Predmet ∗ *zbran1*, IceAge::Predmet ∗ *zbran2*, IceAge::Predmet ∗ *zbran3*)

Konstruktor.

Konstruktor

Definice je uvedena na řádku 9 v souboru Humanoid.cpp.

6.8.2.2 IceAge::Humanoid::~Humanoid ( )

Destruktor.

Destruktor

Definice je uvedena na řádku 42 v souboru Humanoid.cpp.

6.8.3 Dokumentace k metodám

6.8.3.1 short IceAge::Humanoid::obrana ( ) [virtual]

Podedena metoda vracejici hodnotu celkove obrany.

Podedena metoda vracejici hodnotu celkove obrany, coz je odolnost.

Implementuje IceAge::Zivot.

Definice je uvedena na řádku 30 v souboru Humanoid.cpp.

6.8.3.2 short IceAge::Humanoid::utok ( ) [virtual]

Podedena metoda vracejici hodnotu celkoveho utoku.

Podedena metoda vracejici hodnotu celkoveho utoku, coz je sila. Pokud ma v rukach nejake zbrane, prictou se hodnoty i techto zbrani.

Implementuje IceAge::Zivot.

Definice je uvedena na řádku 18 v souboru Humanoid.cpp.

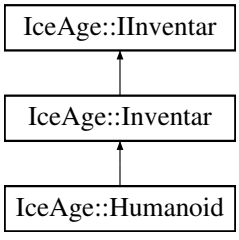
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/Humanoid.h
- cpp/Humanoid.cpp

6.9 Dokumentace třídy IceAge::IInventar

`#include <IInventar.h>`

Diagram dědičnosti pro třídu IceAge::IInventar



Veřejné metody

- virtual void pridejPredmet (IceAge::Predmet *novyPredmet)=0

  Virtualni metoda pridavajici novy predmet do inventare.
- virtual IceAge::Predmet * odeberPredmet ()=0

  Virtualni metoda vyndavajici nejaky predmet z inventare.
- virtual void znicPredmet ()=0

  Virtualni metoda nicici predmet v inventari.

Chráněné metody

- virtual unsigned short getMaximalniVelikost ()=0

  Virtualni metoda vracejici hodnotu maximalni povolene velikosti inventare.
- virtual void setMaximalniVelikost (unsigned short novaMaximalniVelikost)=0

  Virtualni metoda nastavujici maximalni velikost inventare.

6.9.1 Detailní popis

Abstraktni trida, ktera upresnuje rozhrani vsech dedenych trid, tykajicich se Inventare a jeho obsluhy.

Definice je uvedena na řádku 13 v souboru IInventar.h.

6.9.2 Dokumentace k metodám

6.9.2.1 virtual unsigned short IceAge::IInventar::getMaximalniVelikost ( ) [protected], [pure virtual]

Virtualni metoda vracejici hodnotu maximalni povolene velikosti inventare.

Implementováno v IceAge::Inventar.

6.9.2.2 virtual IceAge::Predmet* IceAge::IInventar::odeberPredmet ( ) [pure virtual]

Virtualni metoda vyndavajici nejaky predmet z inventare.

Implementováno v IceAge::Inventar.

6.9.2.3 virtual void IceAge::IInventar::pridejPredmet (IceAge::Predmet * *novyPredmet*) [pure virtual]

Virtualni metoda pridavajici novy predmet do inventare.

Implementováno v IceAge::Inventar.

6.9.2.4 virtual void IceAge::IInventar::setMaximalniVelikost (unsigned short *novaMaximalniVelikost*) [protected], [pure virtual]

Virtualni metoda nastavujici maximalni velikost inventare.

Implementováno v IceAge::Inventar.

6.9.2.5 virtual void IceAge::IInventar::znicPredmet ( ) [pure virtual]

Virtualni metoda nicici predmet v inventari.

Implementováno v IceAge::Inventar.

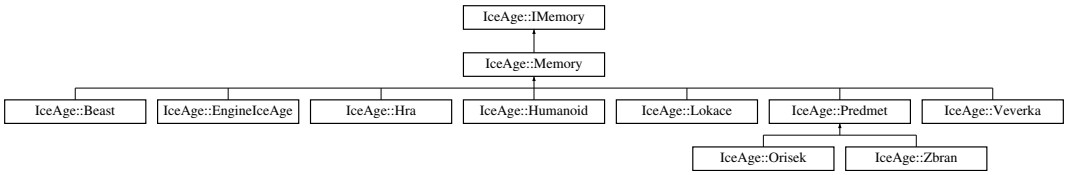
Dokumentace pro tuto třídu byla generována z následujícího souboru:

- hlavicky/IInventar.h

6.10 Dokumentace třídy IceAge::IMemory

`#include <IMemory.h>`

Diagram dědičnosti pro třídu IceAge::IMemory



Veřejné metody

- virtual void setId (unsigned int novaId)=0

  Virtualni metoda nastavujici ID.

Chráněné metody

- virtual unsigned int getId ()=0

  Virtualni metoda, ktera vraci ID (index), na kterem se v MemoryArbiter nachazi.

6.10.1 Detailní popis

Abstratkni trida, ktera upresnuje rozhrani vsech dedenych trid, tykajicich se spravy pameti.

Definice je uvedena na řádku 13 v souboru IMemory.h.

6.10.2 Dokumentace k metodám

6.10.2.1 virtual unsigned int IceAge::IMemory::getId ( ) [protected], [pure virtual]

Virtualni metoda, ktera vraci ID (index), na kterem se v MemoryArbiter nachazi.

Implementováno v IceAge::Memory.

6.10.2.2 virtual void IceAge::IMemory::setId (unsigned int *novaId*) [pure virtual]

Virtualni metoda nastavujici ID.

Implementováno v IceAge::Memory.

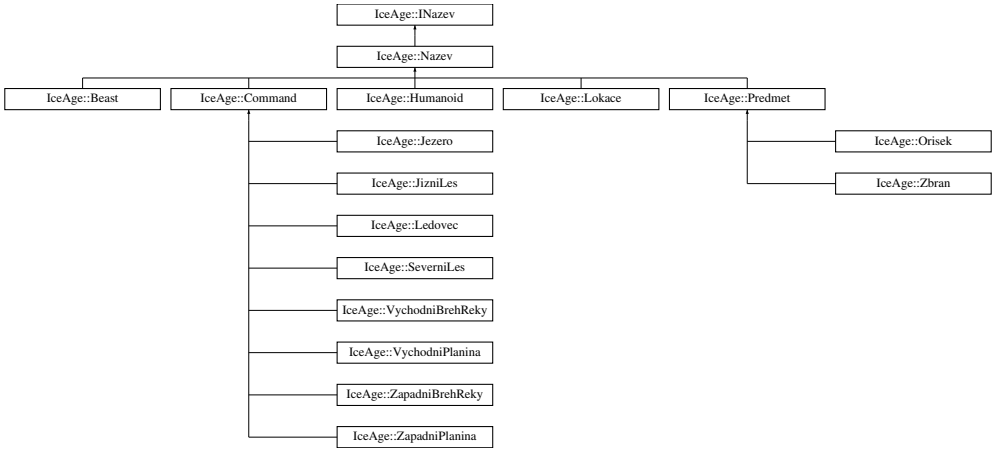
Dokumentace pro tuto třídu byla generována z následujícího souboru:

- hlavicky/IMemory.h

6.11 Dokumentace třídy IceAge::INazev

`#include <INazev.h>`

Diagram dědičnosti pro třídu IceAge::INazev



Veřejné metody

- virtual std::string getNazev ()=0
  Virtualni metoda vracejici nazev.

Chráněné metody

- virtual void setNazev (std::string novyNazev)=0
  Virtualni metoda nastavujici nazev.

6.11.1 Detailní popis

Abstratkni trida, ktera upresnuje rozhrani vsech dedenych trid, tykajicich se nazvu.

Definice je uvedena na řádku 13 v souboru INazev.h.

6.11.2 Dokumentace k metodám

6.11.2.1 virtual std::string IceAge::INazev::getNazev ( ) [pure virtual]

Virtualni metoda vracejici nazev.

Implementováno v IceAge::Nazev.

6.11.2.2 virtual void IceAge::INazev::setNazev (std::string *novyNazev*) [protected], [pure virtual]

Virtualni metoda nastavujici nazev.

Implementováno v IceAge::Nazev.

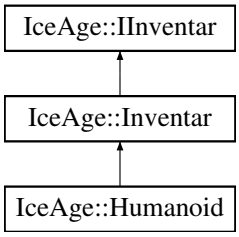
Dokumentace pro tuto třídu byla generována z následujícího souboru:

- hlavicky/INazev.h

6.12 Dokumentace třídy IceAge::Inventar

`#include <Inventar.h>`

Diagram dědičnosti pro třídu IceAge::Inventar



Veřejné metody

- IceAge::Predmet * odeberPredmet ()
  Metoda vyndavajici nejaky predmet z inventare.
- void pridejPredmet (IceAge::Predmet *novyPredmet)
  Metoda pridavajici novy predmet do inventare.
- void znicPredmet ()
  Metoda nicici predmet v inventari.
- void vypisObsah ()
  Metoda vypisujici predmety v inventari.
- Inventar (unsigned short novaVelikost)
- ~Inventar ()

Chráněné metody

- unsigned short getMaximalniVelikost ()

  Metoda vracejici hodnotu maximalni povolene velikosti inventare.
- void setMaximalniVelikost (unsigned short novaMaximalniVelikost)

  Metoda nastavujici maximalni velikost inventare.
- unsigned int volba ()

  Pomocna metoda k rozhodovani.

Chráněné atributy

- std::vector< IceAge::Predmet * > m_vektorPredmetu

  Vektor predmetu fungujici jako inventar.
- unsigned short m_maximalniVelikost

  Atribut udavajici maximalni velikost inventare.

6.12.1 Detailní popis

Abstraktni trida, ktera implementuje rozhrani vsech dedenych trid, tykajicich se Inventare a jeho obsluhy.

Definice je uvedena na řádku 14 v souboru Inventar.h.

6.12.2 Dokumentace konstruktoru a destruktoru

6.12.2.1 IceAge::Inventar::Inventar (unsigned short *novaVelikost*)

Konstruktor, ve kterem se nastavuje maximalni velikost inventare.

Definice je uvedena na řádku 66 v souboru Inventar.cpp.

6.12.2.2 IceAge::Inventar::~Inventar ( )

Destruktor, ve kterem se mazou vsechny predmety v inventari. Probiha kontrola, zda tam neni nullptr.

Definice je uvedena na řádku 73 v souboru Inventar.cpp.

6.12.3 Dokumentace k metodám

6.12.3.1 unsigned short IceAge::Inventar::getMaximalniVelikost ( ) [protected], [virtual]

Metoda vracejici hodnotu maximalni povolene velikosti inventare.

Metoda vracejici hodnotu maximalni povolene velikosti inventare.

Implementuje IceAge::IInventar.

Definice je uvedena na řádku 10 v souboru Inventar.cpp.

6.12.3.2 IceAge::Predmet * IceAge::Inventar::odeberPredmet ( ) [virtual]

Metoda vyndavajici nejaky predmet z inventare.

Metoda vyndavajici nejaky predmet z inventare. Predmet se vybira na zaklade vypisu a volani pomocne metody volba(). V ramci usetreni rychlost se zde meni odebirany prvek s poslednim. Jinak by se shiftovalo.

Implementuje IceAge::IInventar.

Definice je uvedena na řádku 17 v souboru Inventar.cpp.

6.12.3.3 void IceAge::Inventar::pridejPredmet (IceAge::Predmet * *novyPredmet*) [virtual]

Metoda pridavajici novy predmet do inventare.

Metoda pridavajici novy predmet do inventare. Kontroluje se zde maximalni velikost Inventare a pripadne se to zahlasi uzivateli.

Implementuje IceAge::IInventar.

Definice je uvedena na řádku 34 v souboru Inventar.cpp.

6.12.3.4 void IceAge::Inventar::setMaximalniVelikost (unsigned short *novaMaximalniVelikost*) [protected], [virtual]

Metoda nastavujici maximalni velikost inventare.

Metoda nastavujici maximalni velikost inventare. Dane cislo musi byt kladne. Pokud neni, je nastaven inventar na 1.

Implementuje IceAge::IInventar.

Definice je uvedena na řádku 42 v souboru Inventar.cpp.

6.12.3.5 unsigned int IceAge::Inventar::volba ( ) [protected]

Pomocna metoda k rozhodovani.

Pomocna metoda k rozhodovani. Nacita od uzivatele a bude cyklit, dokud to nenacte cislo a zaroven nebude mensi, jak aktualni velikost inventare.

Definice je uvedena na řádku 91 v souboru Inventar.cpp.

6.12.3.6 void IceAge::Inventar::vypisObsah ( )

Metoda vypisujici predmety v inventari.

Metoda vypisujici predmety v inventari.

Definice je uvedena na řádku 81 v souboru Inventar.cpp.

6.12.3.7 void IceAge::Inventar::znicPredmet ( ) [virtual]

Metoda nicici predmet v inventari.

Metoda nicici predmet v inventari. Probiha to na zaklade vypisu a volby. V ramci rychlosti se mazany predmet uklada na konec. Zamezi se tim shiftovani.

Implementuje IceAge::IInventar.

Definice je uvedena na řádku 51 v souboru Inventar.cpp.

6.12.4 Dokumentace k datovým členům

6.12.4.1 unsigned short IceAge::Inventar::m_maximalniVelikost [protected]

Atribut udavajici maximalni velikost inventare.

Definice je uvedena na řádku 18 v souboru Inventar.h.

6.12.4.2 std::vector<IceAge::Predmet*> IceAge::Inventar::m_vektorPredmetu [protected]

Vektor predmetu fungujici jako inventar.

Definice je uvedena na řádku 17 v souboru Inventar.h.

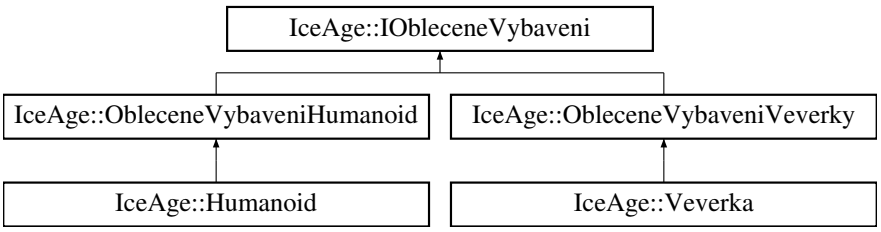
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/Inventar.h
- cpp/Inventar.cpp

6.13 Dokumentace třídy IceAge::IObleceneVybaveni

`#include <IObleceneVybaveni.h>`

Diagram dědičnosti pro třídu IceAge::IObleceneVybaveni



Veřejné metody

- virtual void setLevaRuka (IceAge::Predmet *novaLevaRuka)=0

  Virtualni metoda nastavujici obsah leve ruky.
- virtual void setPravaRuka (IceAge::Predmet *novaPravaRuka)=0

  Virtualni metoda nastavujici obsah prave ruky.

Chráněné metody

- virtual IceAge::Predmet * getLevaRuka ()=0
  Virtualni metoda vracejici obsah leve ruky.
- virtual IceAge::Predmet * getPravaRuka ()=0
  Virtualni metoda vracejici obsah prave ruky.

6.13.1 Detailní popis

Abstratkni trida, ktera upresnuje rozhrani vsech dedenych trid, tykajicich se obleceneho vybaveni (hlavne zbrani).

Definice je uvedena na řádku 13 v souboru IObleceneVybaveni.h.

6.13.2 Dokumentace k metodám

6.13.2.1 virtual IceAge::Predmet* IceAge::IObleceneVybaveni::getLevaRuka ( ) [protected], [pure virtual]

Virtualni metoda vracejici obsah leve ruky.

Implementováno v IceAge::ObleceneVybaveniVeverky a IceAge::ObleceneVybaveniHumanoid.

6.13.2.2 virtual IceAge::Predmet* IceAge::IObleceneVybaveni::getPravaRuka ( ) [protected], [pure virtual]

Virtualni metoda vracejici obsah prave ruky.

Implementováno v IceAge::ObleceneVybaveniVeverky a IceAge::ObleceneVybaveniHumanoid.

6.13.2.3 virtual void IceAge::IObleceneVybaveni::setLevaRuka (IceAge::Predmet * *novaLevaRuka*) [pure virtual]

Virtualni metoda nastavujici obsah leve ruky.

Implementováno v IceAge::ObleceneVybaveniHumanoid a IceAge::ObleceneVybaveniVeverky.

6.13.2.4 virtual void IceAge::IObleceneVybaveni::setPravaRuka (IceAge::Predmet * *novaPravaRuka*) [pure virtual]

Virtualni metoda nastavujici obsah prave ruky.

Implementováno v IceAge::ObleceneVybaveniHumanoid a IceAge::ObleceneVybaveniVeverky.

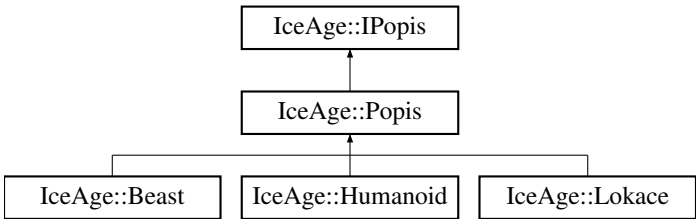
Dokumentace pro tuto třídu byla generována z následujícího souboru:

- hlavicky/IObleceneVybaveni.h

6.14 Dokumentace třídy IceAge::IPopis

`#include <IPopis.h>`

Diagram dědičnosti pro třídu IceAge::IPopis



Chráněné metody

- virtual std::string getPopis ()=0

  Virtualni metoda vracejici popis.
- virtual void setPopis (std::string novyPopis)=0

  Virtualni metoda nastavujici popis.

6.14.1 Detailní popis

Abstratkni trida, ktera upresnuje rozhrani vsech dedenych trid, tykajicich se jakehokoli popisu.

Definice je uvedena na řádku 13 v souboru IPopis.h.

6.14.2 Dokumentace k metodám

6.14.2.1 virtual std::string IceAge::IPopis::getPopis ( ) [protected], [pure virtual]

Virtualni metoda vracejici popis.

Implementováno v IceAge::Popis.

6.14.2.2 virtual void IceAge::IPopis::setPopis (std::string *novyPopis*) [protected], [pure virtual]

Virtualni metoda nastavujici popis.

Implementováno v IceAge::Popis.

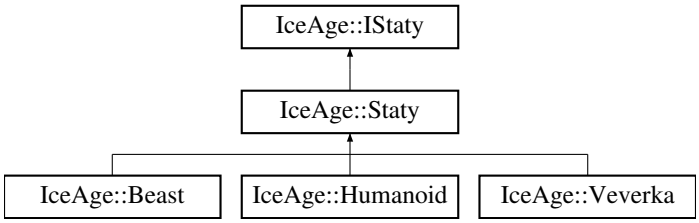
Dokumentace pro tuto třídu byla generována z následujícího souboru:

- hlavicky/IPopis.h

6.15 Dokumentace třídy IceAge::IStaty

`#include <IStaty.h>`

Diagram dědičnosti pro třídu IceAge::IStaty



Chráněné metody

- virtual short getSila ()=0

  Virtualni metoda vracejici hodnotu sily.
- virtual void setSila (short novaSila)=0

  Virtualni metoda nastavujici hodnotu sily.
- virtual short getOdolnost ()=0

  Virtualni metoda vracejici hodnotu odolnosti.
- virtual void setOdolnost (short novaOdolnost)=0

  Virtualni metoda nastavujici hodnotu odolnosti.

6.15.1 Detailní popis

Abstratkni trida, ktera upresnuje rozhrani vsech dedenych trid, tykajicich se zakladnich statu.

Definice je uvedena na řádku 11 v souboru IStaty.h.

6.15.2 Dokumentace k metodám

6.15.2.1 virtual short IceAge::IStaty::getOdolnost ( ) [protected], [pure virtual]

Virtualni metoda vracejici hodnotu odolnosti.

Implementováno v IceAge::Staty.

6.15.2.2 virtual short IceAge::IStaty::getSila ( ) [protected], [pure virtual]

Virtualni metoda vracejici hodnotu sily.

Implementováno v IceAge::Staty.

6.15.2.3 virtual void IceAge::IStaty::setOdolnost (short *novaOdolnost*) [protected], [pure virtual]

Virtualni metoda nastavujici hodnotu odolnosti.

Implementováno v IceAge::Staty.

6.15.2.4 virtual void IceAge::IStaty::setSila (short *novaSila*) [protected], [pure virtual]

Virtualni metoda nastavujici hodnotu sily.

Implementováno v IceAge::Staty.

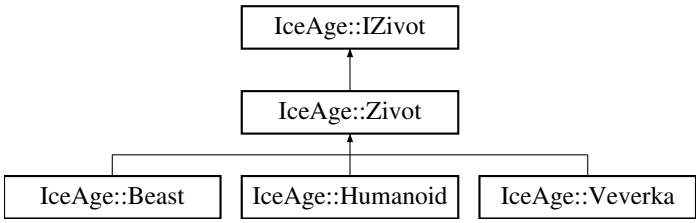
Dokumentace pro tuto třídu byla generována z následujícího souboru:

- hlavicky/IStaty.h

6.16 Dokumentace třídy IceAge::IZivot

`#include <IZivot.h>`

Diagram dědičnosti pro třídu IceAge::IZivot



Veřejné metody

- virtual void changeZivotAktualni (short hodnotaZmeny)=0
  Virtualni metoda menici hodnotu aktualnihoZivota.

Chráněné metody

- virtual short getZivotAktualni ()=0
  Virtualni metoda vracejici hodnotu aktualnihoZivota.
- virtual void setZivotAktualni (short novyZivotAktualni)=0
  Virtualni metoda nastavujici hodnotu aktualnihoZivota.
- virtual short getZivotMaximalni ()=0
  Virtualni metoda vracejici hodnotu maximalnihoZivota.
- virtual void setZivotMaximalni (short novyZivotMaximalni)=0
  Virtualni metoda nastavujici hodnotu maximalnihoZivota.

6.16.1 Detailní popis

Abstratkni trida, ktera upresnuje rozhrani vsech dedenych trid, tykajicich se zivota.

Definice je uvedena na řádku 14 v souboru IZivot.h.

6.16.2 Dokumentace k metodám

6.16.2.1 virtual void IceAge::IZivot::changeZivotAktualni (short *hodnotaZmeny*) [pure virtual]

Virtualni metoda menici hodnotu aktualnihoZivota.

Implementováno v IceAge::Zivot.

6.16.2.2 virtual short IceAge::IZivot::getZivotAktualni ( ) [protected], [pure virtual]

Virtualni metoda vracejici hodnotu aktualnihoZivota.

Implementováno v IceAge::Zivot.

6.16.2.3 virtual short IceAge::IZivot::getZivotMaximalni ( ) [protected], [pure virtual]

Virtualni metoda vracejici hodnotu maximalnihoZivota.

Implementováno v IceAge::Zivot.

6.16.2.4 virtual void IceAge::IZivot::setZivotAktualni (short *novyZivotAktualni*) [protected], [pure virtual]

Virtualni metoda nastavujici hodnotu aktualnihoZivota.

Implementováno v IceAge::Zivot.

6.16.2.5 virtual void IceAge::IZivot::setZivotMaximalni (short *novyZivotMaximalni*) [protected], [pure virtual]

Virtualni metoda nastavujici hodnotu maximalnihoZivota.

Implementováno v IceAge::Zivot.

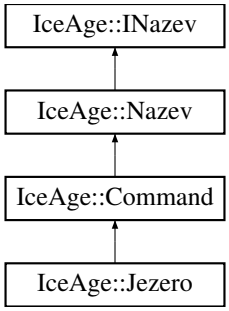
Dokumentace pro tuto třídu byla generována z následujícího souboru:

- hlavicky/IZivot.h

6.17 Dokumentace třídy IceAge::Jezero

`#include <Jezero.h>`

Diagram dědičnosti pro třídu IceAge::Jezero



Veřejné metody

- Jezero (std::string novyNazev)

  Konstruktor.
- ~Jezero ()

  Virtualni Destruktor.
- void prozkoumat (IceAge::Veverka *scrat, std::vector< IceAge::Beast * > vektorBeast, std::vector< IceAge::Humanoid * > vektorHumanoid, IceAge::Reka *reka)

  Podedena metoda pridavajici algoritmus pro prozkoumani lokace Jezero.

Další zděděné členy

6.17.1 Detailní popis

Trida implementujici algoritmus pro prozkoumani lokace Jezero.

Definice je uvedena na řádku 16 v souboru Jezero.h.

6.17.2 Dokumentace konstruktoru a destruktoru

6.17.2.1 IceAge::Jezero::Jezero (std::string *novyNazev*)

Konstruktor.

Konstruktor

Definice je uvedena na řádku 9 v souboru Jezero.cpp.

6.17.2.2 IceAge::Jezero::~Jezero ( )

Virtualni Destruktor.

Destruktor

Definice je uvedena na řádku 16 v souboru Jezero.cpp.

6.17.3 Dokumentace k metodám

6.17.3.1 void IceAge::Jezero::prozkoumat (IceAge::Veverka * *scrat*, std::vector< IceAge::Beast * > *vektorBeast*, std::vector< IceAge::Humanoid * > *vektorHumanoid*, IceAge::Reka * *reka*) [virtual]

Podedena metoda pridavajici algoritmus pro prozkoumani lokace Jezero.

Podedena metoda pridavajici algoritmus pro prozkoumani lokace Jezero. <Lokalni promenna pozivana pro nacitani odpovedi od uzivatele.

<Lokalni promenna pouzivana pro cislovani moznosti, ktere se zobrazi uzivateli.

Vykresleni obrazku

<Lokalni promenna pouzivana pro udrzovani cyklu v chodu. Konkretne u ziskavani odpovedi od uzivatele.

<Cyklus pro pasaz cmuchani.

<Reset lokalnich promennych na puvodni hodnoty.

<Cyklus pro hledani dal.

<Reset lokalnich promennych na puvodni hodnoty.

Pokud ma hrac stale lodicku, tak mu ji sebereme.

<Lokalni promenna pouzivana pro udrzovani cyklu v chodu. Konkretne u opakovani vypisu nejake cinnosti a ziskavani odpovedi (cyklus2)

<Cyklus pro trhani proutku. Pokud neutrhnu vic kusu, budu tu cyklit.

<Ziskavani odpovedi od uzivatele.

<Reset lokalnich promennych na puvodni hodnoty.

<Reset lokalnich promennych na puvodni hodnoty.

Pasaz se Supomutim mladetem

Hrac ziskal orisek a utika pryc.

<Cyklus pro ziskani odpovedi "kudy zdrhnout"

<Reset lokalnich promennych na puvodni hodnoty.

<Zahozeni klacku.

Pasaz s rybickama a kung-fu Scratem

<Cyklus pro ziskani odpovedi a pripadne pro boj s rybami

Zde je popsan boj s rybami. Boj je zcela automaticky a casovany. Pred bojem se hracovi zvednou staty

Zaverecny obrazek a vypis

<Uvolneni pameti.

<Uspesne ukonceni.

Implementuje IceAge::Command.

Definice je uvedena na řádku 22 v souboru Jezero.cpp.

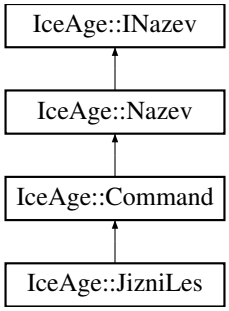
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/Jezero.h
- cpp/Jezero.cpp

6.18 Dokumentace třídy IceAge::JizniLes

`#include <JizniLes.h>`

Diagram dědičnosti pro třídu IceAge::JizniLes



Veřejné metody

- JizniLes (std::string novyNazev)

  Konstruktor.
- ~JizniLes ()

  Virtualni Destruktor.
- void prozkoumat (IceAge::Veverka *scrat, std::vector< IceAge::Beast * > vektorBeast, std::vector< IceAge::Humanoid * > vektorHumanoid, IceAge::Reka *reka)

  Podedena metoda pridavajici algoritmus pro prozkoumani lokace JizniLes.

Další zděděné členy

6.18.1 Detailní popis

Trida implementujici algoritmus pro prozkoumani lokace JizniLes.

Definice je uvedena na řádku 13 v souboru JizniLes.h.

6.18.2 Dokumentace konstruktoru a destruktoru

6.18.2.1 IceAge::JizniLes::JizniLes (std::string *novyNazev*)

Konstruktor.

Konstruktor

Definice je uvedena na řádku 9 v souboru JizniLes.cpp.

6.18.2.2 IceAge::JizniLes::~JizniLes ( )

Virtualni Destruktor.

Destruktor

Definice je uvedena na řádku 15 v souboru JizniLes.cpp.

6.18.3 Dokumentace k metodám

6.18.3.1 void IceAge::JizniLes::prozkoumat (IceAge::Veverka * *scrat*, std::vector< IceAge::Beast * > *vektorBeast*, std::vector< IceAge::Humanoid * > *vektorHumanoid*, IceAge::Reka * *reka*) [virtual]

Podedena metoda pridavajici algoritmus pro prozkoumani lokace JizniLes.

Podedena metoda pridavajici algoritmus pro prozkoumani lokace JizniLes. Vykresleni obrazku

<Lokalni promenna pouzivana pro udrzovani cyklu v chodu. Konkretne u ziskavani odpovedi od uzivatele.

<Lokalni promenna pouzivana pro cislovani moznosti, ktere se zobrazi uzivateli.

<Lokalni promenna pouzivana pro vybrani odpovedi, musi byt schopna dosahnout nejvyssich hodnot indexu vectoru.

<Lokalni promenna pozivana pro nacitani odpovedi od uzivatele.

<Cyklus pro hlavni nabidku prozkoumani.

<Moznost odejit

<Moznost napit/uzdravit se

<Moznost preplavani bez lodky

<Moznost zautocit na nekoho

<Vybrani na koho se utoci

<Moznost sebrani klacku

<Moznost sebrani lodicky

Implementuje IceAge::Command.

Definice je uvedena na řádku 22 v souboru JizniLes.cpp.

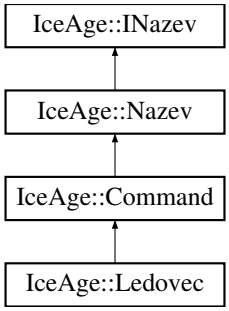
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/JizniLes.h
- cpp/JizniLes.cpp

6.19 Dokumentace třídy IceAge::Ledovec

`#include <Ledovec.h>`

Diagram dědičnosti pro třídu IceAge::Ledovec



Veřejné metody

- Ledovec (std::string novyNazev)

  Konstruktor.
- ~Ledovec ()

  Virtualni Destruktor.
- void prozkoumat (IceAge::Veverka *scrat, std::vector< IceAge::Beast * > vektorBeast, std::vector< IceAge::Humanoid * > vektorHumanoid, IceAge::Reka *reka)

  Podedena metoda pridavajici algoritmus pro prozkoumani lokace Ledovec.

Další zděděné členy

6.19.1 Detailní popis

Trida implementujici algoritmus pro prozkoumani lokace Ledovec.

Definice je uvedena na řádku 14 v souboru Ledovec.h.

6.19.2 Dokumentace konstruktoru a destruktoru

6.19.2.1 IceAge::Ledovec::Ledovec (std::string *novyNazev*)

Konstruktor.

Konstruktor

Definice je uvedena na řádku 9 v souboru Ledovec.cpp.

6.19.2.2 IceAge::Ledovec::~Ledovec ( )

Virtualni Destruktor.

Destruktor

Definice je uvedena na řádku 15 v souboru Ledovec.cpp.

6.19.3 Dokumentace k metodám

6.19.3.1 void IceAge::Ledovec::prozkoumat (IceAge::Veverka * *scrat*, std::vector< IceAge::Beast * > *vektorBeast*, std::vector< IceAge::Humanoid * > *vektorHumanoid*, IceAge::Reka * *reka*) [virtual]

Podedena metoda pridavajici algoritmus pro prozkoumani lokace Ledovec.

Podedena metoda pridavajici algoritmus pro prozkoumani lokace Ledovec. <Sebere se zivot

<Ukaze kdo trefil hrace

Implementuje IceAge::Command.

Definice je uvedena na řádku 21 v souboru Ledovec.cpp.

Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/Ledovec.h
- cpp/Ledovec.cpp

6.20 Dokumentace třídy IceAge::LocationDirector

`#include <LocationDirector.h>`

Veřejné metody

- LocationDirector (IceAge::BuilderLokace *novyBuilder)
  Konstruktor.
- ~LocationDirector ()
  Destruktor.
- IceAge::Lokace * createLokace ()
  Metoda ktera vytvori Lokace.
- void setBuilder (IceAge::BuilderLokace *novyBuilder)
  Metoda ktera nastavuje BuilderLokace.

Privátní atributy

- IceAge::BuilderLokace * m_builder
  Atribut udrzujici odkaz na BuilderLokace.

6.20.1 Detailní popis

Troda ktera ridi vytvareni lokaci.

Definice je uvedena na řádku 13 v souboru LocationDirector.h.

6.20.2 Dokumentace konstruktoru a destruktoru

6.20.2.1 IceAge::LocationDirector::LocationDirector (IceAge::BuilderLokace * *novyBuilder*)

Konstruktor.

Konstruktor nastavujici BuilderLokace.

Definice je uvedena na řádku 9 v souboru LocationDirector.cpp.

6.20.2.2 IceAge::LocationDirector::~LocationDirector ( )

Destruktor.

Destruktor ve kterem se maze builder.

Definice je uvedena na řádku 31 v souboru LocationDirector.cpp.

6.20.3 Dokumentace k metodám

6.20.3.1 IceAge::Lokace * IceAge::LocationDirector::createLokace ( )

Metoda ktera vytvori Lokace.

Metoda ktera vytvori Lokace a vrati aktualni.

Definice je uvedena na řádku 16 v souboru LocationDirector.cpp.

6.20.3.2 void IceAge::LocationDirector::setBuilder (IceAge::BuilderLokace * *novyBuilder*)

Metoda ktera nastavuje BuilderLokace.

Metoda ktera nastavuje BuilderLokace

Definice je uvedena na řádku 24 v souboru LocationDirector.cpp.

6.20.4 Dokumentace k datovým členům

6.20.4.1 IceAge::BuilderLokace* IceAge::LocationDirector::m_builder [private]

Atribut udrzujici odkaz na BuilderLokace.

Definice je uvedena na řádku 16 v souboru LocationDirector.h.

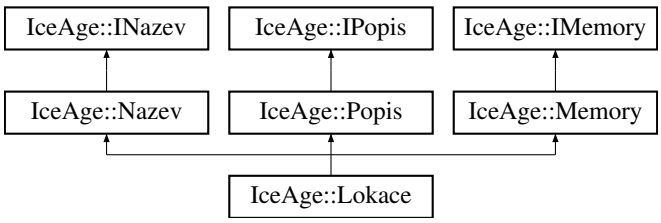
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/LocationDirector.h
- cpp/LocationDirector.cpp

6.21 Dokumentace třídy IceAge::Lokace

```
#include <Lokace.h>
```

Diagram dědičnosti pro třídu IceAge::Lokace



Veřejné metody

- Lokace (std::string novyNazev, std::string novyPopis)
- ~Lokace ()
- void pridejBeast (IceAge::Beast *novyBeast)
- void pridejHumanoid (IceAge::Humanoid *novyHumanoid)
- void smazBeast (unsigned int index)
- void smazHumanoid (unsigned int index)
- void pridejReku ()
- void smazReku ()
- void pridejLokaci (IceAge::Lokace *novaLokace)
- void smazLokaci (unsigned int index)
- void odeberLokaci (unsigned int index)
- void vypisLokace ()
- std::string getNazevLokace ()
- std::string getPopisLokace ()
- Lokace * getLokace (unsigned int index)
- IceAge::Beast * getBeast (unsigned int index)
- IceAge::Humanoid * getHumanoid (unsigned int index)
- unsigned int getVelikostVektoruLokaci ()
- std::vector< IceAge::Beast * > getVektorBeast ()
- std::vector< IceAge::Humanoid * > getVektorHumanoid ()
- IceAge::Reka * getReka ()
- void setPrikaz ()
- void prozkoumatLokaci (IceAge::Veverka *scrat)

Privátní atributy

- std::vector< IceAge::Beast * > m_vektorBeast
- std::vector< IceAge::Humanoid * > m_vektorHumanoid
- IceAge::Reka * m_odkazReka
- std::vector< IceAge::Lokace * > m_vektorLokaci
- IceAge::Command * m_prikaz

Další zděděné členy

6.21.1 Detailní popis

Definice je uvedena na řádku 34 v souboru Lokace.h.

6.21.2 Dokumentace konstruktoru a destruktoru

6.21.2.1 IceAge::Lokace::Lokace (std::string *novyNazev,* std::string *novyPopis*)

Definice je uvedena na řádku 6 v souboru Lokace.cpp.

6.21.2.2 IceAge::Lokace::~Lokace ( )

Definice je uvedena na řádku 101 v souboru Lokace.cpp.

6.21.3 Dokumentace k metodám

6.21.3.1 IceAge::Beast * IceAge::Lokace::getBeast (unsigned int *index*)

Definice je uvedena na řádku 79 v souboru Lokace.cpp.

6.21.3.2 IceAge::Humanoid * IceAge::Lokace::getHumanoid (unsigned int *index*)

Definice je uvedena na řádku 84 v souboru Lokace.cpp.

6.21.3.3 IceAge::Lokace * IceAge::Lokace::getLokace (unsigned int *index*)

Definice je uvedena na řádku 115 v souboru Lokace.cpp.

6.21.3.4 std::string IceAge::Lokace::getNazevLokace ( )

Definice je uvedena na řádku 71 v souboru Lokace.cpp.

6.21.3.5 std::string IceAge::Lokace::getPopisLokace ( )

Definice je uvedena na řádku 75 v souboru Lokace.cpp.

6.21.3.6 IceAge::Reka * IceAge::Lokace::getReka ( )

Definice je uvedena na řádku 132 v souboru Lokace.cpp.

6.21.3.7 std::vector< IceAge::Beast * > IceAge::Lokace::getVektorBeast ( )

Definice je uvedena na řádku 124 v souboru Lokace.cpp.

6.21.3.8 std::vector< IceAge::Humanoid * > IceAge::Lokace::getVektorHumanoid ( )

Definice je uvedena na řádku 128 v souboru Lokace.cpp.

6.21.3.9 unsigned int IceAge::Lokace::getVelikostVektoruLokaci ( )

Definice je uvedena na řádku 67 v souboru Lokace.cpp.

6.21.3.10 void IceAge::Lokace::odeberLokaci (unsigned int *index*)

Definice je uvedena na řádku 53 v souboru Lokace.cpp.

6.21.3.11 void IceAge::Lokace::pridejBeast (IceAge::Beast * *novyBeast*)

Definice je uvedena na řádku 11 v souboru Lokace.cpp.

6.21.3.12 void IceAge::Lokace::pridejHumanoid (IceAge::Humanoid * *novyHumanoid*)

Definice je uvedena na řádku 15 v souboru Lokace.cpp.

6.21.3.13 void IceAge::Lokace::pridejLokaci (IceAge::Lokace * *novaLokace*)

Definice je uvedena na řádku 41 v souboru Lokace.cpp.

6.21.3.14 void IceAge::Lokace::pridejReku ( )

Definice je uvedena na řádku 33 v souboru Lokace.cpp.

6.21.3.15 void IceAge::Lokace::prozkoumatLokaci (IceAge::Veverka * *scrat*)

Definice je uvedena na řádku 120 v souboru Lokace.cpp.

6.21.3.16 void IceAge::Lokace::setPrikaz ( )

Definice je uvedena na řádku 89 v souboru Lokace.cpp.

6.21.3.17 void IceAge::Lokace::smazBeast (unsigned int *index*)

Definice je uvedena na řádku 19 v souboru Lokace.cpp.

6.21.3.18 void IceAge::Lokace::smazHumanoid (unsigned int *index*)

Definice je uvedena na řádku 26 v souboru Lokace.cpp.

6.21.3.19 void IceAge::Lokace::smazLokaci (unsigned int *index*)

Definice je uvedena na řádku 45 v souboru Lokace.cpp.

6.21.3.20 void IceAge::Lokace::smazReku ( )

Definice je uvedena na řádku 37 v souboru Lokace.cpp.

6.21.3.21 void IceAge::Lokace::vypisLokace ( )

Definice je uvedena na řádku 60 v souboru Lokace.cpp.

6.21.4 Dokumentace k datovým členům

6.21.4.1 IceAge::Reka* IceAge::Lokace::m_odkazReka [private]

Definice je uvedena na řádku 39 v souboru Lokace.h.

6.21.4.2 IceAge::Command* IceAge::Lokace::m_prikaz [private]

Definice je uvedena na řádku 41 v souboru Lokace.h.

6.21.4.3 std::vector<IceAge::Beast*> IceAge::Lokace::m_vektorBeast [private]

Definice je uvedena na řádku 37 v souboru Lokace.h.

6.21.4.4 std::vector<IceAge::Humanoid*> IceAge::Lokace::m_vektorHumanoid [private]

Definice je uvedena na řádku 38 v souboru Lokace.h.

6.21.4.5 std::vector<IceAge::Lokace*> IceAge::Lokace::m_vektorLokaci [private]

Definice je uvedena na řádku 40 v souboru Lokace.h.

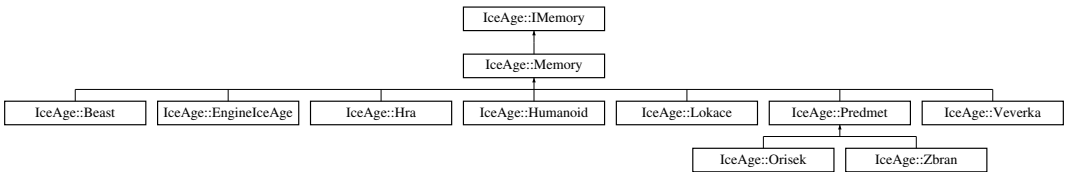
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/Lokace.h
- cpp/Lokace.cpp

6.22 Dokumentace třídy IceAge::Memory

`#include <Memory.h>`

Diagram dědičnosti pro třídu IceAge::Memory



Veřejné metody

- void setId (unsigned int noveId)

  Nastavuje hodnotu ID.
- Memory ()

  Konstruktor.
- virtual ~Memory ()

  Destruktor.

Chráněné metody

- unsigned int getId ()

  Vraci hodnotu ID.

Chráněné atributy

- unsigned int m_id

  ID (index), na kterem se v MemoryArbiter nachazi dedena trida.

6.22.1 Detailní popis

Abstratkni trida, ktera implementuje rozhrani vsech dedenych trid, tykajicich se spravy pameti. Musi se z ni dedit, pokud ma byt jina trida ulozena v MemoryArbiter.

Definice je uvedena na řádku 20 v souboru Memory.h.

6.22.2 Dokumentace konstruktoru a destruktoru

6.22.2.1 IceAge::Memory::Memory ( )

Konstruktor.

Konstruktor, ve kterem se vola metoda MemoryArbitera, pridavajici sam sebe do jeho seznamu.

Definice je uvedena na řádku 23 v souboru Memory.cpp.

6.22.2.2 IceAge::Memory::~Memory ( ) [virtual]

Destruktor.

Destruktor, ve kterem se vola metoda MemoryArbitera, odebirajici sam sebe z jeho seznamu.

Definice je uvedena na řádku 30 v souboru Memory.cpp.

6.22.3 Dokumentace k metodám

6.22.3.1 unsigned int IceAge::Memory::getId ( ) [protected], [virtual]

Vraci hodnotu ID.

Vraci hodnotu ID.

Implementuje IceAge::IMemory.

Definice je uvedena na řádku 9 v souboru Memory.cpp.

6.22.3.2 void IceAge::Memory::setId (unsigned int *noveId*) [virtual]

Nastavuje hodnotu ID.

Nastavuje hodnotu ID. Hodnota, ktera se sem vklada, bude vzdy kladna (jedna se o index vectoru).

Implementuje IceAge::IMemory.

Definice je uvedena na řádku 16 v souboru Memory.cpp.

6.22.4 Dokumentace k datovým členům

6.22.4.1 unsigned int IceAge::Memory::m_id [protected]

ID (index), na kterem se v MemoryArbiter nachazi dedena trida.

Definice je uvedena na řádku 23 v souboru Memory.h.

Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/Memory.h
- cpp/Memory.cpp

6.23 Dokumentace třídy IceAge::MemoryArbiter

`#include <MemoryArbiter.h>`

Statické veřejné metody

- static void add (IceAge::Memory *target)

  Staticka metoda pridavajici odkaz na tridu do memoryVector.
- static void remove (unsigned int targetId)

  Staticka metoda odebirajici odkaz na tridu z memoryVector.
- static void cycleDelete ()

  Staticka metoda uvolnujici veskerou pamet, na kterou ma odkaz.

Privátní metody

- MemoryArbiter ()

  Privatni Konstruktor, ktery se nikdy nezavola.

Statické privátní atributy

- static std::vector< IceAge::Memory * > s_memoryVector

  Staticky vektor udrzujici odkazy na jednotlive tridy.

6.23.1 Detailní popis

Staticka Trida, obsluhujici pamet. Vsechny dulezite tridy (zabirajici vice mista nebo obsahujici vice jinych trid v ramci agregace/kompozice) zde na sebe ukladaji odkaz. Pokud ma tato trida fungovat, musi si hlidana trida podedit abstraktni tridu Memory. V pripade, ze nedojde k smazani dane tridy v programu, MemoryArbiter muze sam uvolnit jeji pamet. Problem muze nastat v pripade dosazeni maximalniho poctu prvku v memoryVector. Trida je implementovana jako Monostate.

Definice je uvedena na řádku 22 v souboru MemoryArbiter.h.

6.23.2 Dokumentace konstruktoru a destruktoru

6.23.2.1 IceAge::MemoryArbiter::MemoryArbiter ( ) [private]

Privatni Konstruktor, ktery se nikdy nezavola.

Konstruktor, ktery se nikdy nebude volat.

Definice je uvedena na řádku 12 v souboru MemoryArbiter.cpp.

6.23.3 Dokumentace k metodám

6.23.3.1 void IceAge::MemoryArbiter::add (IceAge::Memory * *target*) [static]

Staticka metoda pridavajici odkaz na tridu do memoryVector.

Staticka metoda pridavajici instanci nejake tridy do vectoru memoryVector. Nasledne zavola metodu setId() z prave pridane tridy a tim ji ulozi jeji ID v seznamu.

Definice je uvedena na řádku 18 v souboru MemoryArbiter.cpp.

6.23.3.2 void IceAge::MemoryArbiter::cycleDelete ( ) [static]

Staticka metoda uvolnujici veskerou pamet, na kterou ma odkaz.

Staticka metoda, ktera smaze kompletni seznam uchovavane pameti v memoryVector. <Musi se kontrolovat, zda se tam nenechazi nullptr. Nektere instance mohou v destruktorech smazat jine instance.

Definice je uvedena na řádku 35 v souboru MemoryArbiter.cpp.

6.23.3.3 void IceAge::MemoryArbiter::remove (unsigned int *targetId*) [static]

Staticka metoda odebirajici odkaz na tridu z memoryVector.

Staticka metoda odebirajici mazanou instanci tridy ze seznamu. Provadi to ve vlastnim destrutkoru => neni potreba tu uchovavat odkaz a zabirat misto. <Nejprve se vynuluje odkaz mazane instance.

Definice je uvedena na řádku 26 v souboru MemoryArbiter.cpp.

6.23.4 Dokumentace k datovým členům

6.23.4.1 std::vector< IceAge::Memory * > IceAge::MemoryArbiter::s_memoryVector [static], [private]

Staticky vektor udrzujici odkazy na jednotlive tridy.

Oznamuje, ze se bude tento vector v programu vyskytovat. Musi byt v CPP souboru, v hlavickem to haze errory.

Definice je uvedena na řádku 25 v souboru MemoryArbiter.h.

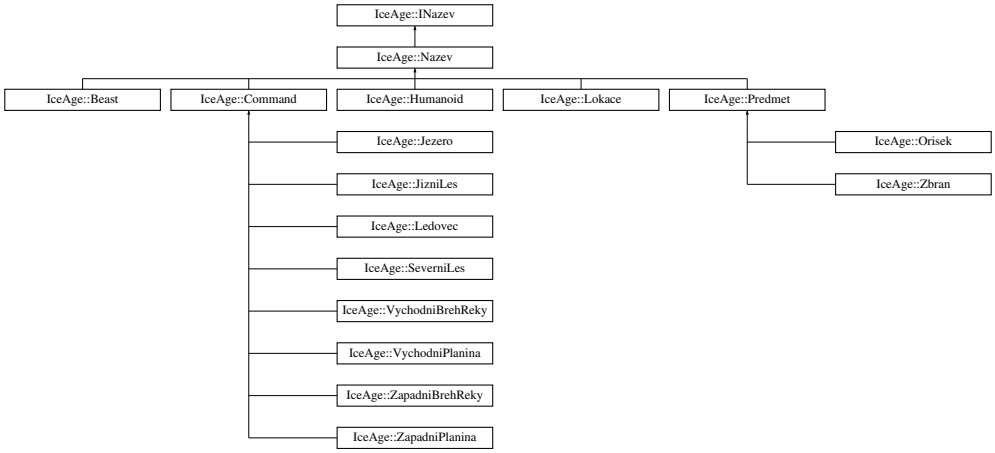
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/MemoryArbiter.h
- cpp/MemoryArbiter.cpp

6.24 Dokumentace třídy IceAge::Nazev

`#include <Nazev.h>`

Diagram dědičnosti pro třídu IceAge::Nazev



Veřejné metody

- Nazev (std::string novyNazev)

  Konstruktor.
- std::string getNazev ()

  Metoda vracejici nazev.

Chráněné metody

- void setNazev (std::string novyNazev)

  Metoda nastavujici nazev.

Chráněné atributy

- std::string m_nazev

  Atribut do ktereho se uklada nazev dedene tridy (jeji instance).

6.24.1 Detailní popis

Abstratkni trida, ktera implementuje rozhrani vsech dedenych trid, tykajicich se nazvu.

Definice je uvedena na řádku 14 v souboru Nazev.h.

6.24.2 Dokumentace konstruktoru a destruktoru

6.24.2.1 IceAge::Nazev::Nazev (std::string *novyNazev*)

Konstruktor.

Konstruktor, ve kterem se nastavuje nazev.

Definice je uvedena na řádku 24 v souboru Nazev.cpp.

6.24.3 Dokumentace k metodám

6.24.3.1 std::string IceAge::Nazev::getNazev ( ) [virtual]

Metoda vracejici nazev.

Metoda vracejici nazev.

Implementuje IceAge::INazev.

Definice je uvedena na řádku 10 v souboru Nazev.cpp.

6.24.3.2 void IceAge::Nazev::setNazev (std::string *novyNazev*) [protected], [virtual]

Metoda nastavujici nazev.

Metoda nastavujici nazev. Nazev neni v tuto chvili nijak omezen.

Implementuje IceAge::INazev.

Definice je uvedena na řádku 17 v souboru Nazev.cpp.

6.24.4 Dokumentace k datovým členům

6.24.4.1 std::string IceAge::Nazev::m_nazev [protected]

Atribut do ktereho se uklada nazev dedene tridy (jeji instance).

Definice je uvedena na řádku 17 v souboru Nazev.h.

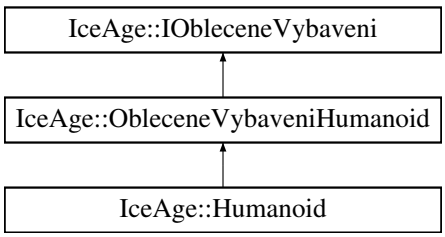
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/Nazev.h
- cpp/Nazev.cpp

6.25 Dokumentace třídy IceAge::ObleceneVybaveniHumanoid

`#include <ObleceneVybaveniHumanoid.h>`

Diagram dědičnosti pro třídu IceAge::ObleceneVybaveniHumanoid



Veřejné metody

- void setZada (IceAge::Predmet ∗novaZada)

  Metoda nastavujici obsah zad.
- void setPravaRuka (IceAge::Predmet ∗novaPravaRuka)

  Metoda nastavujici obsah prave ruky.
- void setLevaRuka (IceAge::Predmet ∗novaLevaRuka)

  Metoda nastavujici obsah leve ruky.
- ObleceneVybaveniHumanoid (IceAge::Predmet ∗levaRuka, IceAge::Predmet ∗pravaRuka, IceAge::Predmet ∗zada)

  Konstruktor.
- ∼ObleceneVybaveniHumanoid ()

  Destruktor.

Chráněné metody

- IceAge::Predmet ∗ getLevaRuka ()

  Metoda vracejici obsah leve ruky.
- IceAge::Predmet ∗ getPravaRuka ()

  Metoda vracejici obsah prave ruky.
- IceAge::Predmet ∗ getZada ()

  Metoda vracejici obsah zad.

Chráněné atributy

- IceAge::Predmet ∗ m_levaRuka

  Atribut udrzujici odkaz na predmet v leve ruce.
- IceAge::Predmet ∗ m_pravaRuka

  Atribut udrzujici odkaz na predmet v prave ruce.
- IceAge::Predmet ∗ m_zada

  Atribut udrzujici odkaz na predmet na zadech.

6.25.1 Detailní popis

Abstratkni trida, ktera implementuje rozhrani vsech dedenych trid, tykajicich se obleceneho vybaveni (hlavne zbrani). Konkretne pro dve ruce a zada.

Definice je uvedena na řádku 14 v souboru ObleceneVybaveniHumanoid.h.

6.25.2 Dokumentace konstruktoru a destruktoru

6.25.2.1 IceAge::ObleceneVybaveniHumanoid::ObleceneVybaveniHumanoid (IceAge::Predmet * *levaRuka,* IceAge::Predmet * *pravaRuka,* IceAge::Predmet * *zada*)

Konstruktor.

Konstruktor ve kterem se nastavuji nove predmety do rukou a zad.

Definice je uvedena na řádku 51 v souboru ObleceneVybaveniHumanoid.cpp.

6.25.2.2 IceAge::ObleceneVybaveniHumanoid::~ObleceneVybaveniHumanoid ( )

Destruktor.

Destruktor, ve kterem se mazou predmety z rukou a zad.

Definice je uvedena na řádku 60 v souboru ObleceneVybaveniHumanoid.cpp.

6.25.3 Dokumentace k metodám

6.25.3.1 IceAge::Predmet * IceAge::ObleceneVybaveniHumanoid::getLevaRuka ( ) [protected], [virtual]

Metoda vracejici obsah leve ruky.

Metoda vracejici obsah leve ruky.

Implementuje IceAge::IObleceneVybaveni.

Definice je uvedena na řádku 9 v souboru ObleceneVybaveniHumanoid.cpp.

6.25.3.2 IceAge::Predmet * IceAge::ObleceneVybaveniHumanoid::getPravaRuka ( ) [protected], [virtual]

Metoda vracejici obsah prave ruky.

Metoda vracejici obsah prave ruky.

Implementuje IceAge::IObleceneVybaveni.

Definice je uvedena na řádku 23 v souboru ObleceneVybaveniHumanoid.cpp.

6.25.3.3 IceAge::Predmet * IceAge::ObleceneVybaveniHumanoid::getZada ( ) [protected]

Metoda vracejici obsah zad.

Metoda nastavujici obsah zad. Neni kladeno zadne omezeni.

Definice je uvedena na řádku 37 v souboru ObleceneVybaveniHumanoid.cpp.

6.25.3.4 void IceAge::ObleceneVybaveniHumanoid::setLevaRuka (IceAge::Predmet * *novaLevaRuka*) [virtual]

Metoda nastavujici obsah leve ruky.

Metoda nastavujici obsah leve ruky. Neni kladeno zadne omezeni.

Implementuje IceAge::IObleceneVybaveni.

Definice je uvedena na řádku 16 v souboru ObleceneVybaveniHumanoid.cpp.

6.25.3.5 void IceAge::ObleceneVybaveniHumanoid::setPravaRuka (IceAge::Predmet * *novaPravaRuka*) [virtual]

Metoda nastavujici obsah prave ruky.

Metoda nastavujici obsah prave ruky. Neni kladeno zadne omezeni.

Implementuje IceAge::IObleceneVybaveni.

Definice je uvedena na řádku 30 v souboru ObleceneVybaveniHumanoid.cpp.

6.25.3.6 void IceAge::ObleceneVybaveniHumanoid::setZada (IceAge::Predmet * *novaZada*)

Metoda nastavujici obsah zad.

Metoda nastavujici obsah zad. Neni kladeno zadne omezeni.

Definice je uvedena na řádku 44 v souboru ObleceneVybaveniHumanoid.cpp.

6.25.4 Dokumentace k datovým členům

6.25.4.1 IceAge::Predmet* IceAge::ObleceneVybaveniHumanoid::m_levaRuka [protected]

Atribut udrzujici odkaz na predmet v leve ruce.

Definice je uvedena na řádku 17 v souboru ObleceneVybaveniHumanoid.h.

6.25.4.2 IceAge::Predmet* IceAge::ObleceneVybaveniHumanoid::m_pravaRuka [protected]

Atribut udrzujici odkaz na predmet v prave ruce.

Definice je uvedena na řádku 18 v souboru ObleceneVybaveniHumanoid.h.

6.25.4.3 IceAge::Predmet* IceAge::ObleceneVybaveniHumanoid::m_zada [protected]

Atribut udrzujici odkaz na predmet na zadech.

Definice je uvedena na řádku 19 v souboru ObleceneVybaveniHumanoid.h.

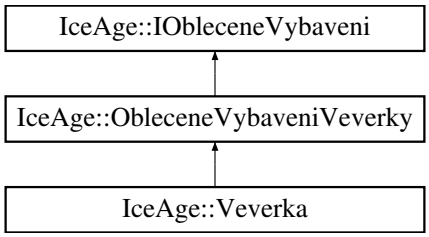
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/ObleceneVybaveniHumanoid.h
- cpp/ObleceneVybaveniHumanoid.cpp

6.26 Dokumentace třídy IceAge::ObleceneVybaveniVeverky

`#include <ObleceneVybaveniVeverky.h>`

Diagram dědičnosti pro třídu IceAge::ObleceneVybaveniVeverky



Veřejné metody

- void setPravaRuka (IceAge::Predmet *novaPravaRuka)
  Metoda nastavujici obsah prave ruky.
- void setLevaRuka (IceAge::Predmet *novaLevaRuka)
  Metoda nastavujici obsah leve ruky.
- ObleceneVybaveniVeverky (IceAge::Predmet *novaLevaRuka, IceAge::Predmet *novaPravaRuka)
  Konstruktor.
- ~ObleceneVybaveniVeverky ()
  Destruktor.
- IceAge::Predmet * getLevaRuka ()
  Metoda vracejici obsah leve ruky.
- IceAge::Predmet * getPravaRuka ()
  Metoda vracejici obsah prave ruky.
- void zahodLevaRuka ()
  Metoda zahazujici (mazani) predmet v leve ruce.
- void zahodPravaRuka ()
  Metoda zahazujici (mazani) predmet v prave ruce.

Chráněné atributy

- IceAge::Predmet * m_levaRuka
  Atribut udrzujici odkaz na predmet v leve ruce.
- IceAge::Predmet * m_pravaRuka
  Atribut udrzujici odkaz na predmet v prave ruce.

Další zděděné členy

6.26.1 Detailní popis

Abstratkni trida, ktera implementuje rozhrani vsech dedenych trid, tykajicich se obleceneho vybaveni (hlavne zbrani). Konkretne pro dve ruce.

Definice je uvedena na řádku 16 v souboru ObleceneVybaveniVeverky.h.

6.26.2 Dokumentace konstruktoru a destruktoru

6.26.2.1 IceAge::ObleceneVybaveniVeverky::ObleceneVybaveniVeverky (IceAge::Predmet * *novaLevaRuka*, IceAge::Predmet * *novaPravaRuka*)

Konstruktor.

Konstruktor ve kterem se nastavuji nove predmety do rukou.

Definice je uvedena na řádku 37 v souboru ObleceneVybaveniVeverky.cpp.

6.26.2.2 IceAge::ObleceneVybaveniVeverky::~ObleceneVybaveniVeverky ( )

Destruktor.

Destruktor, ve kterem se mazou predmety z rukou.

Definice je uvedena na řádku 45 v souboru ObleceneVybaveniVeverky.cpp.

6.26.3 Dokumentace k metodám

6.26.3.1 IceAge::Predmet * IceAge::ObleceneVybaveniVeverky::getLevaRuka ( ) [virtual]

Metoda vracejici obsah leve ruky.

Metoda vracejici obsah leve ruky.

Implementuje IceAge::IObleceneVybaveni.

Definice je uvedena na řádku 9 v souboru ObleceneVybaveniVeverky.cpp.

6.26.3.2 IceAge::Predmet * IceAge::ObleceneVybaveniVeverky::getPravaRuka ( ) [virtual]

Metoda vracejici obsah prave ruky.

Metoda vracejici obsah prave ruky.

Implementuje IceAge::IObleceneVybaveni.

Definice je uvedena na řádku 23 v souboru ObleceneVybaveniVeverky.cpp.

6.26.3.3 void IceAge::ObleceneVybaveniVeverky::setLevaRuka (IceAge::Predmet * *novaLevaRuka*) [virtual]

Metoda nastavujici obsah leve ruky.

Metoda nastavujici obsah leve ruky. Neni kladeno zadne omezeni.

Implementuje IceAge::IObleceneVybaveni.

Definice je uvedena na řádku 16 v souboru ObleceneVybaveniVeverky.cpp.

6.26.3.4 void IceAge::ObleceneVybaveniVeverky::setPravaRuka (IceAge::Predmet * *novaPravaRuka*) [virtual]

Metoda nastavujici obsah prave ruky.

Metoda nastavujici obsah prave ruky. Neni kladeno zadne omezeni.

Implementuje IceAge::IObleceneVybaveni.

Definice je uvedena na řádku 30 v souboru ObleceneVybaveniVeverky.cpp.

6.26.3.5 void IceAge::ObleceneVybaveniVeverky::zahodLevaRuka ( )

Metoda zahazujici (mazani) predmet v leve ruce.

Metoda zahazujici (mazani) predmet v leve ruce. Nasledne nastavi atribut na nullptr.

Definice je uvedena na řádku 53 v souboru ObleceneVybaveniVeverky.cpp.

6.26.3.6 void IceAge::ObleceneVybaveniVeverky::zahodPravaRuka ( )

Metoda zahazujici (mazani) predmet v prave ruce.

Metoda zahazujici (mazani) predmet v prave ruce. Nasledne nastavi atribut na nullptr.

Definice je uvedena na řádku 61 v souboru ObleceneVybaveniVeverky.cpp.

6.26.4 Dokumentace k datovým členům

6.26.4.1 IceAge::Predmet* IceAge::ObleceneVybaveniVeverky::m_levaRuka [protected]

Atribut udrzujici odkaz na predmet v leve ruce.

Definice je uvedena na řádku 19 v souboru ObleceneVybaveniVeverky.h.

6.26.4.2 IceAge::Predmet* IceAge::ObleceneVybaveniVeverky::m_pravaRuka [protected]

Atribut udrzujici odkaz na predmet v prave ruce.

Definice je uvedena na řádku 20 v souboru ObleceneVybaveniVeverky.h.

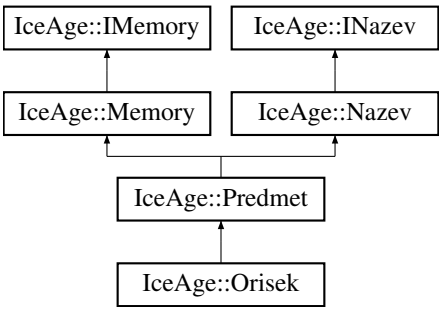
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/ObleceneVybaveniVeverky.h
- cpp/ObleceneVybaveniVeverky.cpp

6.27 Dokumentace třídy IceAge::Orisek

`#include <Orisek.h>`

Diagram dědičnosti pro třídu IceAge::Orisek



Veřejné metody

- Orisek (std::string novyPopis)

  Konstruktor.
- ~Orisek ()

  Destruktor.
- short getAtribut ()

  Podedena metoda vracejici hodnotu 0.

Další zděděné členy

6.27.1 Detailní popis

Trida, ktera znazornuje predmet typu Orisek a implementuje pro nej rozhrani. Tato trida je urcena pro jakykoli predmet, ktery potrebuje pouze nazev a nic jineho.

Definice je uvedena na řádku 13 v souboru Orisek.h.

6.27.2 Dokumentace konstruktoru a destruktoru

6.27.2.1 IceAge::Orisek::Orisek (std::string *novyNazev*)

Konstruktor.

Konstruktor

Definice je uvedena na řádku 9 v souboru Orisek.cpp.

6.27.2.2 IceAge::Orisek::~Orisek ( )

Destruktor.

Destruktor

Definice je uvedena na řádku 16 v souboru Orisek.cpp.

6.27.3 Dokumentace k metodám

6.27.3.1 short IceAge::Orisek::getAtribut ( ) [virtual]

Podedena metoda vracejici hodnotu 0.

Podedena metoda vracejici hodnotu 0.

Implementuje IceAge::Predmet.

Definice je uvedena na řádku 22 v souboru Orisek.cpp.

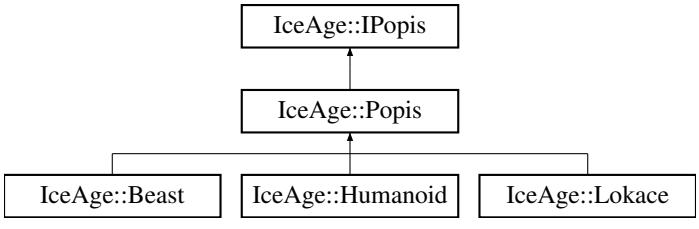
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/Orisek.h
- cpp/Orisek.cpp

6.28 Dokumentace třídy IceAge::Popis

```
#include <Popis.h>
```

Diagram dědičnosti pro třídu IceAge::Popis



Veřejné metody

- Popis (std::string novyPopis)

  Konstruktor.
- ~Popis ()

  Destruktor.
- std::string getPopis ()

  Metoda vracejici popis.

Chráněné metody

- void setPopis (std::string novyPopis)

  Metoda nastavujici popis.

Chráněné atributy

- std::string m_popis

  Atribut do ktereho se uklada popis.

6.28.1 Detailní popis

Abstratkni trida, ktera upresnuje rozhrani vsech dedenych trid, tykajicich se jakehokoli popisu.

Definice je uvedena na řádku 13 v souboru Popis.h.

6.28.2 Dokumentace konstruktoru a destruktoru

6.28.2.1 IceAge::Popis::Popis (std::string *novyPopis*)

Konstruktor.

Konstruktor ve kterem se nastavuje popis.

Definice je uvedena na řádku 23 v souboru Popis.cpp.

6.28.2.2 IceAge::Popis::~Popis ( )

Destruktor.

Destruktor

Definice je uvedena na řádku 30 v souboru Popis.cpp.

6.28.3 Dokumentace k metodám

6.28.3.1 std::string IceAge::Popis::getPopis ( ) [virtual]

Metoda vracejici popis.

Metoda vracejici popis.

Implementuje IceAge::IPopis.

Definice je uvedena na řádku 9 v souboru Popis.cpp.

6.28.3.2 void IceAge::Popis::setPopis (std::string *novyPopis*) [protected], [virtual]

Metoda nastavujici popis.

Metoda nastavujici popis. Neni zde zadne omezeni.

Implementuje IceAge::IPopis.

Definice je uvedena na řádku 16 v souboru Popis.cpp.

6.28.4 Dokumentace k datovým členům

6.28.4.1 std::string IceAge::Popis::m_popis [protected]

Atribut do ktereho se uklada popis.

Definice je uvedena na řádku 16 v souboru Popis.h.

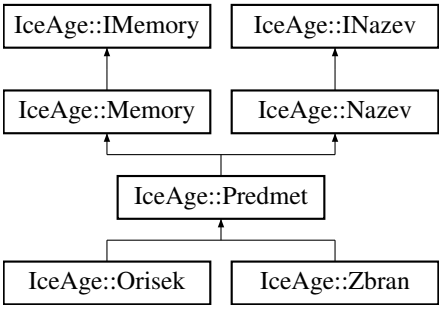
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/Popis.h
- cpp/Popis.cpp

6.29 Dokumentace třídy IceAge::Predmet

```
#include <Predmet.h>
```

Diagram dědičnosti pro třídu IceAge::Predmet



Veřejné metody

- char getTyp ()

  Metoda vracejici typ(druh) predmetu.
- virtual short getAtribut ()=0

  Virtualni metoda vracejici hodnotu atributu nejakeho potomka.
- Predmet (std::string novyNazev)

  Konstruktor.
- ~Predmet ()

  Destruktor.

Chráněné metody

- void setTyp (char novyTyp)

  Metoda nastavujici typ.

Chráněné atributy

- char m_typ

  Atribut uchovavajici hodnotu, ktera znazornuje o ktery predmet se jedna. 'w' je zbran, 'n' je zbytek, 'u' je nezname.

6.29.1 Detailní popis

Abstraktni trida, ktera znazornuje predmet a castecne pro nej implementuje rozhrani.

Definice je uvedena na řádku 15 v souboru Predmet.h.

6.29.2 Dokumentace konstruktoru a destruktoru

6.29.2.1 IceAge::Predmet::Predmet (std::string *novyNazev*)

Konstruktor.

Konstruktor

Definice je uvedena na řádku 23 v souboru Predmet.cpp.

6.29.2.2 IceAge::Predmet::~Predmet ( )

Destruktor.

Destruktor

Definice je uvedena na řádku 29 v souboru Predmet.cpp.

6.29.3 Dokumentace k metodám

6.29.3.1 virtual short IceAge::Predmet::getAtribut ( ) [pure virtual]

Virtualni metoda vracejici hodnotu atributu nejakeho potomka.

Implementováno v IceAge::Zbran a IceAge::Orisek.

6.29.3.2 char IceAge::Predmet::getTyp ( )

Metoda vracejici typ(druh) predmetu.

Metoda vracejici typ(druh) predmetu.

Definice je uvedena na řádku 8 v souboru Predmet.cpp.

6.29.3.3 void IceAge::Predmet::setTyp (char *novyTyp*) [protected]

Metoda nastavujici typ.

Metoda nastavujici typ. Ma omezeni na tri hodnoty.

Definice je uvedena na řádku 15 v souboru Predmet.cpp.

6.29.4 Dokumentace k datovým členům

6.29.4.1 char IceAge::Predmet::m_typ [protected]

Atribut uchovavajici hodnotu, ktera znazornuje o ktery predmet se jedna. 'w' je zbran, 'n' je zbytek, 'u' je nezname.

Definice je uvedena na řádku 18 v souboru Predmet.h.

Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/Predmet.h
- cpp/Predmet.cpp

6.30 Dokumentace třídy IceAge::Reka

```
#include <Reka.h>
```

Veřejné metody

- short getHodnotaZivota ()

  Metoda vracejici hodnotaZivota.
- Reka (short novaHodnota)

  Konstruktor.
- ~Reka ()

  Destruktor.

Privátní metody

- void setHodnotaZivota (short novaHodnotaZivota)

  Metoda nastavujici hodnotaZivota.

Privátní atributy

- short m_hodnotaZivota

  Atribut udavajici o kolik se uzdravis.

6.30.1 Detailní popis

Trida, implementujici rozhrani pro praci s rekou.

Definice je uvedena na řádku 11 v souboru Reka.h.

6.30.2 Dokumentace konstruktoru a destruktoru

6.30.2.1 IceAge::Reka::Reka (short *novaHodnota*)

Konstruktor.

Konstruktor

Definice je uvedena na řádku 23 v souboru Reka.cpp.

6.30.2.2 IceAge::Reka::~Reka ( )

Destruktor.

Destruktor

Definice je uvedena na řádku 30 v souboru Reka.cpp.

6.30.3 Dokumentace k metodám

6.30.3.1 short IceAge::Reka::getHodnotaZivota ( )

Metoda vracejici hodnotaZivota.

Metoda vracejici hodnotaZivota.

Definice je uvedena na řádku 9 v souboru Reka.cpp.

6.30.3.2 void IceAge::Reka::setHodnotaZivota (short *novaHodnotaZivota*) [private]

Metoda nastavujici hodnotaZivota.

Metoda nastavujici hodnotaZivota.

Definice je uvedena na řádku 16 v souboru Reka.cpp.

6.30.4 Dokumentace k datovým členům

6.30.4.1 short IceAge::Reka::m_hodnotaZivota [private]

Atribut udavajici o kolik se uzdravis.

Definice je uvedena na řádku 14 v souboru Reka.h.

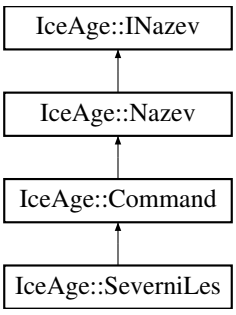
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/Reka.h
- cpp/Reka.cpp

6.31 Dokumentace třídy IceAge::SeverniLes

`#include <SeverniLes.h>`

Diagram dědičnosti pro třídu IceAge::SeverniLes



Veřejné metody

- SeverniLes (std::string novyNazev)

  Konstruktor.
- ~SeverniLes ()

  Virtualni Destruktor.
- void prozkoumat (IceAge::Veverka *scrat, std::vector< IceAge::Beast * > vektorBeast, std::vector< IceAge::Humanoid * > vektorHumanoid, IceAge::Reka *reka)

  Podedena metoda pridavajici algoritmus pro prozkoumani lokace SeverniLes.

Další zděděné členy

6.31.1 Detailní popis

Trida implementujici algoritmus pro prozkoumani lokace SeverniLes.

Definice je uvedena na řádku 13 v souboru SeverniLes.h.

6.31.2 Dokumentace konstruktoru a destruktoru

6.31.2.1 IceAge::SeverniLes::SeverniLes (std::string *novyNazev*)

Konstruktor.

Konstruktor

Definice je uvedena na řádku 9 v souboru SeverniLes.cpp.

6.31.2.2 IceAge::SeverniLes::~SeverniLes ( )

Virtualni Destruktor.

Destruktor

Definice je uvedena na řádku 15 v souboru SeverniLes.cpp.

6.31.3 Dokumentace k metodám

6.31.3.1 void IceAge::SeverniLes::prozkoumat (IceAge::Veverka * *scrat*, std::vector< IceAge::Beast * > *vektorBeast*, std::vector< IceAge::Humanoid * > *vektorHumanoid*, IceAge::Reka * *reka*) [virtual]

Podedena metoda pridavajici algoritmus pro prozkoumani lokace SeverniLes.

Podedena metoda pridavajici algoritmus pro prozkoumani lokace SeverniLes. <Lokalni promenna pouzivana pro udrzovani cyklu v chodu. Konkretne u ziskavani odpovedi od uzivatele.

<Lokalni promenna pouzivana pro cislovani moznosti, ktere se zobrazi uzivateli.

<Lokalni promenna pozivana pro nacitani odpovedi od uzivatele.

<Cyklus hlavni moznosti prozkoumani.

<Moznost nenapadne se vytratit

<Moznost napit/uzdravit se

Implementuje IceAge::Command.

Definice je uvedena na řádku 21 v souboru SeverniLes.cpp.

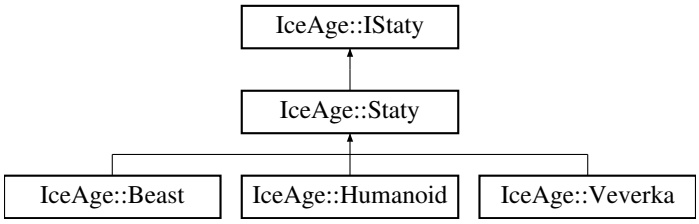
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/SeverniLes.h
- cpp/SeverniLes.cpp

6.32 Dokumentace třídy IceAge::Staty

`#include <Staty.h>`

Diagram dědičnosti pro třídu IceAge::Staty



Veřejné metody

- Staty (short novaSila, short novaOdolnost)

  Konstruktor.
- ~Staty ()

  Destruktor.

Chráněné metody

- short getSila ()

  Metoda vracejici hodnotu sily.
- void setSila (short novaSila)

  Metoda nastavujici hodnotu sily.
- short getOdolnost ()

  Metoda vracejici hodnotu odolnosti.
- void setOdolnost (short novaOdolnost)

  Metoda nastavujici hodnotu odolnosti.

Chráněné atributy

- short m_sila

  Atribut do ktereho se uklada sila.
- short m_odolnost

  Atribut do ktereho se uklada odolnost.

6.32.1 Detailní popis

Abstratkni trida, ktera implementuje rozhrani vsech dedenych trid, tykajicich se zakladnich statu.

Definice je uvedena na řádku 14 v souboru Staty.h.

6.32.2 Dokumentace konstruktoru a destruktoru

6.32.2.1 IceAge::Staty::Staty (short *novaSila,* short *novaOdolnost*)

Konstruktor.

Konstruktor, ve kterem se nastavuje sila i odolnost.

Definice je uvedena na řádku 37 v souboru Staty.cpp.

6.32.2.2 IceAge::Staty::~Staty ( )

Destruktor.

Destruktor

Definice je uvedena na řádku 45 v souboru Staty.cpp.

6.32.3 Dokumentace k metodám

6.32.3.1 short IceAge::Staty::getOdolnost ( ) [protected], [virtual]

Metoda vracejici hodnotu odolnosti.

Metoda vracejici hodnotu odolnosti.

Implementuje IceAge::IStaty.

Definice je uvedena na řádku 23 v souboru Staty.cpp.

6.32.3.2 short IceAge::Staty::getSila ( ) [protected], [virtual]

Metoda vracejici hodnotu sily.

Metoda vracejici hodnotu sily.

Implementuje IceAge::IStaty.

Definice je uvedena na řádku 9 v souboru Staty.cpp.

6.32.3.3 void IceAge::Staty::setOdolnost (short *novaOdolnost*) [protected], [virtual]

Metoda nastavujici hodnotu odolnosti.

Metoda nastavujici hodnotu odolnosti. Muze obsahovat i zaporne hodnoty.

Implementuje IceAge::IStaty.

Definice je uvedena na řádku 30 v souboru Staty.cpp.

6.32.3.4 void IceAge::Staty::setSila (short *novaSila*) [protected], [virtual]

Metoda nastavujici hodnotu sily.

Metoda nastavujici hodnotu sily. Muze obsahovat i zaporne hodnoty.

Implementuje IceAge::IStaty.

Definice je uvedena na řádku 16 v souboru Staty.cpp.

6.32.4 Dokumentace k datovým členům

6.32.4.1 short IceAge::Staty::m_odolnost [protected]

Atribut do ktereho se uklada odolnost.

Definice je uvedena na řádku 18 v souboru Staty.h.

6.32.4.2 short IceAge::Staty::m_sila [protected]

Atribut do ktereho se uklada sila.

Definice je uvedena na řádku 17 v souboru Staty.h.

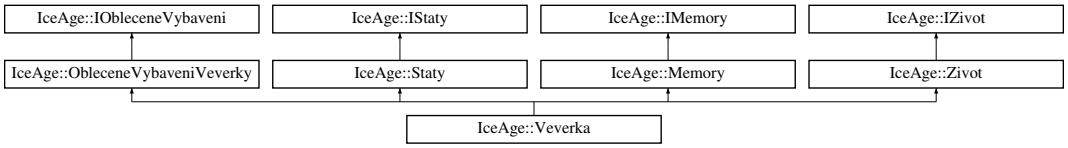
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/Staty.h
- cpp/Staty.cpp

6.33 Dokumentace třídy IceAge::Veverka

`#include <Veverka.h>`

Diagram dědičnosti pro třídu IceAge::Veverka



Veřejné metody

- short utok ()

  Konkretni implementace dedene metody vracejici hodnotu celkoveho utoku.
- short obrana ()

  Konkretni implementace dedene metody vracejici hodnotu celkove obrany.
- ~Veverka ()

  Destruktor.
- void enrageZapnout ()

  Okamzite zvysuje atributy na mnohem vyssi hodnoty.
- void enrageVypnout ()

  Vraci atributy do normalnich hodnot.

Statické veřejné metody

- static IceAge::Veverka * getInstance ()

  Staticka metoda, ktera povoli vytvorit pouze jedinou instanci teto tridy.

Privátní metody

- Veverka (short novyZivot, short novaSila, short novaOdolnost)

  Konstruktor.

Statické privátní atributy

- static IceAge::Veverka * s_vlastniInstance = nullptr

  Staticky atribut odkaz sam na sebe. Dulezite pro Singleton.

Další zděděné členy

6.33.1 Detailní popis

Trida hlavniho hrdiny. Je implementovana jako Singleton.

Definice je uvedena na řádku 21 v souboru Veverka.h.

6.33.2 Dokumentace konstruktoru a destruktoru

6.33.2.1 IceAge::Veverka::Veverka (short *novyZivot,* short *novaSila,* short *novaOdolnost*) [private]

Konstruktor.

Konstruktor, ve kterem se nastavuje novy predmet Orisek. Je privatni, coz zarucuje, ze pujde vytvaret instance pouze ze staticke metody.

Definice je uvedena na řádku 11 v souboru Veverka.cpp.

6.33.2.2 IceAge::Veverka::~Veverka ( )

Destruktor.

Destruktor

Definice je uvedena na řádku 50 v souboru Veverka.cpp.

6.33.3 Dokumentace k metodám

6.33.3.1 void IceAge::Veverka::enrageVypnout ( )

Vraci atributy do normalnich hodnot.

Vraci atributy do normalnich hodnot. Jedna se o silu a odolnost.

Definice je uvedena na řádku 65 v souboru Veverka.cpp.

6.33.3.2 void IceAge::Veverka::enrageZapnout ( )

Okamzite zvysuje atributy na mnohem vyssi hodnoty.

Okamzite zvysuje atributy na mnohem vyssi hodnoty. Jedna se o silu a odolnost.

Definice je uvedena na řádku 56 v souboru Veverka.cpp.

6.33.3.3 IceAge::Veverka * IceAge::Veverka::getInstance ( ) [static]

Staticka metoda, ktera povoli vytvorit pouze jedinou instanci teto tridy.

Staticka metoda, ktera povoli vytvorit pouze jedinou instanci teto tridy. Pokud jiz instance existuje, vrati odkaz na ni misto vytvareni nove.

Definice je uvedena na řádku 19 v souboru Veverka.cpp.

6.33.3.4 short IceAge::Veverka::obrana ( ) [virtual]

Konkretni implementace dedene metody vracejici hodnotu celkove obrany.

Konkretni implementace dedene metody vracejici hodnotu celkove obrany, coz je odolnost.

Implementuje IceAge::Zivot.

Definice je uvedena na řádku 38 v souboru Veverka.cpp.

6.33.3.5 short IceAge::Veverka::utok ( ) [virtual]

Konkretni implementace dedene metody vracejici hodnotu celkoveho utoku.

Konkretni implementace dedene metody vracejici hodnotu celkoveho utoku, coz je sila. Pokud nese veverka i zbrane, pripocita se i jejich poskozeni.

Implementuje IceAge::Zivot.

Definice je uvedena na řádku 27 v souboru Veverka.cpp.

6.33.4 Dokumentace k datovým členům

6.33.4.1 IceAge::Veverka * IceAge::Veverka::s_vlastniInstance = nullptr [static], [private]

Staticky atribut odkaz sam na sebe. Dulezite pro Singleton.

Inicializace statickeho atributu.

Definice je uvedena na řádku 24 v souboru Veverka.h.

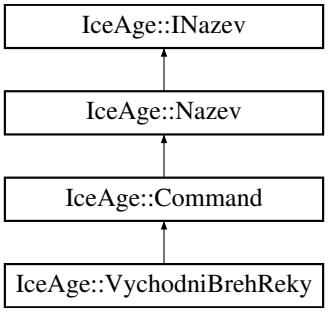
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/Veverka.h
- cpp/Veverka.cpp

6.34 Dokumentace třídy IceAge::VychodniBrehReky

`#include <VychodniBrehReky.h>`

Diagram dědičnosti pro třídu IceAge::VychodniBrehReky



Veřejné metody

- VychodniBrehReky (std::string novyNazev)

  Konstruktor.
- ~VychodniBrehReky ()

  Virtualni Destruktor.
- void prozkoumat (IceAge::Veverka *scrat, std::vector< IceAge::Beast * > vektorBeast, std::vector< IceAge::Humanoid * > vektorHumanoid, IceAge::Reka *reka)

  Podedena metoda pridavajici algoritmus pro prozkoumani lokace VychodniBrehReky.

Další zděděné členy

6.34.1 Detailní popis

Trida implementujici algoritmus pro prozkoumani lokace VychodniBrehReky.

Definice je uvedena na řádku 13 v souboru VychodniBrehReky.h.

6.34.2 Dokumentace konstruktoru a destruktoru

6.34.2.1 IceAge::VychodniBrehReky::VychodniBrehReky (std::string *novyNazev*)

Konstruktor.

Konstruktor

Definice je uvedena na řádku 9 v souboru VychodniBrehReky.cpp.

6.34.2.2 IceAge::VychodniBrehReky::~VychodniBrehReky ( )

Virtualni Destruktor.

Destruktor

Definice je uvedena na řádku 15 v souboru VychodniBrehReky.cpp.

6.34.3 Dokumentace k metodám

6.34.3.1 void IceAge::VychodniBrehReky::prozkoumat (IceAge::Veverka * *scrat*, std::vector< IceAge::Beast * > *vektorBeast*, std::vector< IceAge::Humanoid * > *vektorHumanoid*, IceAge::Reka * *reka*) [virtual]

Podedena metoda pridavajici algoritmus pro prozkoumani lokace VychodniBrehReky.

Podedena metoda pridavajici algoritmus pro prozkoumani lokace VychodniBrehReky. <Lokalni promenna pouzivana pro udrzovani cyklu v chodu. Konkretne u ziskavani odpovedi od uzivatele.

<Lokalni promenna pouzivana pro cislovani moznosti, ktere se zobrazi uzivateli.

<Lokalni promenna pouzivana pro urcovani koho pujde hrac zmlatit.

<Lokalni promenna pozivana pro nacitani odpovedi od uzivatele.

<Cyklus hlavnich moznosti prozkoumavani.

<Moznost odejit.

<Moznost napit/uzdravit se.

<Moznost preplavat reku bez lodky.

<Moznost zautocit na nekoho.

<Vybrani si koho pujde zmlatit.

<Nulovani lokalni promenne.

<Moznost hledat Orisek Cichem

Implementuje IceAge::Command.

Definice je uvedena na řádku 21 v souboru VychodniBrehReky.cpp.

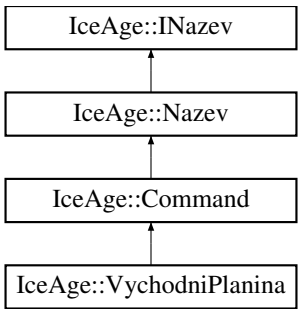
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/VychodniBrehReky.h
- cpp/VychodniBrehReky.cpp

6.35 Dokumentace třídy IceAge::VychodniPlanina

`#include <VychodniPlanina.h>`

Diagram dědičnosti pro třídu IceAge::VychodniPlanina



Veřejné metody

- VychodniPlanina (std::string novyNazev)

  Konstruktor.
- ~VychodniPlanina ()

  Virtualni Destruktor.
- void prozkoumat (IceAge::Veverka *scrat, std::vector< IceAge::Beast * > vektorBeast, std::vector< IceAge::Humanoid * > vektorHumanoid, IceAge::Reka *reka)

  Podedena metoda pridavajici algoritmus pro prozkoumani lokace VychodniPlanina.

Další zděděné členy

6.35.1 Detailní popis

Trida implementujici algoritmus pro prozkoumani lokace VychodniPlanina.

Definice je uvedena na řádku 13 v souboru VychodniPlanina.h.

6.35.2 Dokumentace konstruktoru a destruktoru

6.35.2.1 IceAge::VychodniPlanina::VychodniPlanina (std::string *novyNazev*)

Konstruktor.

Konstruktor

Definice je uvedena na řádku 9 v souboru VychodniPlanina.cpp.

6.35.2.2 IceAge::VychodniPlanina::~VychodniPlanina ( )

Virtualni Destruktor.

Destruktor

Definice je uvedena na řádku 15 v souboru VychodniPlanina.cpp.

6.35.3 Dokumentace k metodám

6.35.3.1 void IceAge::VychodniPlanina::prozkoumat (IceAge::Veverka * *scrat*, std::vector< IceAge::Beast * > *vektorBeast*, std::vector< IceAge::Humanoid * > *vektorHumanoid*, IceAge::Reka * *reka*) [virtual]

Podedena metoda pridavajici algoritmus pro prozkoumani lokace VychodniPlanina.

Podedena metoda pridavajici algoritmus pro prozkoumani lokace VychodniPlanina. <Lokalni promenna pouzivana pro udrzovani cyklu v chodu. Konkretne u ziskavani odpovedi od uzivatele.

<Lokalni promenna pouzivana pro cislovani moznosti, ktere se zobrazi uzivateli.

<Lokalni promenna pozivana pro nacitani odpovedi od uzivatele.

<Cyklus hlavnich moznosti prozkoumavani.

<Moznost odejit.

Implementuje IceAge::Command.

Definice je uvedena na řádku 21 v souboru VychodniPlanina.cpp.

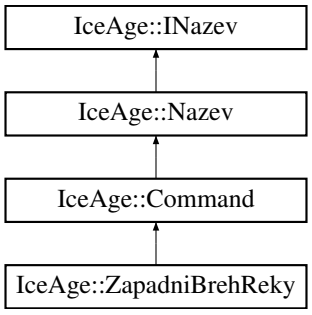
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/VychodniPlanina.h
- cpp/VychodniPlanina.cpp

6.36 Dokumentace třídy IceAge::ZapadniBrehReky

```
#include <ZapadniBrehReky.h>
```

Diagram dědičnosti pro třídu IceAge::ZapadniBrehReky



Veřejné metody

- ZapadniBrehReky (std::string novyNazev)

  Konstruktor.
- ~ZapadniBrehReky ()

  Virtualni Destruktor.
- void prozkoumat (IceAge::Veverka *scrat, std::vector< IceAge::Beast * > vektorBeast, std::vector< IceAge::Humanoid * > vektorHumanoid, IceAge::Reka *reka)

  Podedena metoda pridavajici algoritmus pro prozkoumani lokace ZapadniBrehReky.

Další zděděné členy

6.36.1 Detailní popis

Trida implementujici algoritmus pro prozkoumani lokace ZapadniBrehReky.

Definice je uvedena na řádku 14 v souboru ZapadniBrehReky.h.

6.36.2 Dokumentace konstruktoru a destruktoru

6.36.2.1 IceAge::ZapadniBrehReky::ZapadniBrehReky (std::string *novyNazev*)

Konstruktor.

Konstruktor

Definice je uvedena na řádku 9 v souboru ZapadniBrehReky.cpp.

6.36.2.2 IceAge::ZapadniBrehReky::~ZapadniBrehReky ( )

Virtualni Destruktor.

Destruktor

Definice je uvedena na řádku 15 v souboru ZapadniBrehReky.cpp.

6.36.3 Dokumentace k metodám

6.36.3.1 void IceAge::ZapadniBrehReky::prozkoumat (IceAge::Veverka * *scrat*, std::vector< IceAge::Beast * > *vektorBeast*, std::vector< IceAge::Humanoid * > *vektorHumanoid*, IceAge::Reka * *reka*) [virtual]

Podedena metoda pridavajici algoritmus pro prozkoumani lokace ZapadniBrehReky.

Podedena metoda pridavajici algoritmus pro prozkoumani lokace ZapadniBrehReky. <Lokalni promenna pouzivana pro udrzovani cyklu v chodu. Konkretne u ziskavani odpovedi od uzivatele.

<Lokalni promenna pouzivana pro cislovani moznosti, ktere se zobrazi uzivateli.

<Lokalni promenna pouzivana pro urcovani koho pujde hrac zmlatit.

<Lokalni promenna pozivana pro nacitani odpovedi od uzivatele.

<Cyklus hlavnich moznosti prozkoumavani.

<Moznost odejit.

<Moznost napit/uzdravit se.

<Moznost preplavat reku bez lodky.

<Moznost zautocit na nekoho.

<Vybrani si koho pujde zmlatit.

<Nulovani lokalni promenne.

Implementuje IceAge::Command.

Definice je uvedena na řádku 21 v souboru ZapadniBrehReky.cpp.

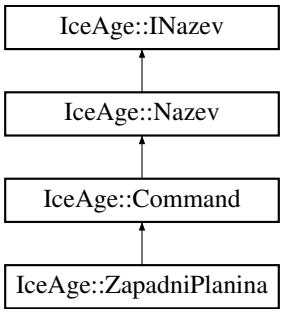
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/ZapadniBrehReky.h
- cpp/ZapadniBrehReky.cpp

6.37 Dokumentace třídy IceAge::ZapadniPlanina

```
#include <ZapadniPlanina.h>
```

Diagram dědičnosti pro třídu IceAge::ZapadniPlanina



Veřejné metody

- ZapadniPlanina (std::string novyNazev)

  Konstruktor.
- ~ZapadniPlanina ()

  Virtualni Destruktor.
- void prozkoumat (IceAge::Veverka *scrat, std::vector< IceAge::Beast * > vektorBeast, std::vector< IceAge::Humanoid * > vektorHumanoid, IceAge::Reka *reka)

  Podedena metoda pridavajici algoritmus pro prozkoumani lokace ZapadniPlanina.

Další zděděné členy

6.37.1 Detailní popis

Trida implementujici algoritmus pro prozkoumani lokace ZapadniPlanina.

Definice je uvedena na řádku 13 v souboru ZapadniPlanina.h.

6.37.2 Dokumentace konstruktoru a destruktoru

6.37.2.1 IceAge::ZapadniPlanina::ZapadniPlanina (std::string *novyNazev*)

Konstruktor.

Konstruktor

Definice je uvedena na řádku 9 v souboru ZapadniPlanina.cpp.

6.37.2.2 IceAge::ZapadniPlanina::~ZapadniPlanina ( )

Virtualni Destruktor.

Destruktor

Definice je uvedena na řádku 15 v souboru ZapadniPlanina.cpp.

6.37.3 Dokumentace k metodám

6.37.3.1 void IceAge::ZapadniPlanina::prozkoumat (IceAge::Veverka * *scrat*, std::vector< IceAge::Beast * > *vektorBeast*, std::vector< IceAge::Humanoid * > *vektorHumanoid*, IceAge::Reka * *reka*) [virtual]

Podedena metoda pridavajici algoritmus pro prozkoumani lokace ZapadniPlanina.

Podedena metoda pridavajici algoritmus pro prozkoumani lokace ZapadniPlanina. Zde se provadi jen vypis toho co se tam odehrava.

Implementuje IceAge::Command.

Definice je uvedena na řádku 21 v souboru ZapadniPlanina.cpp.

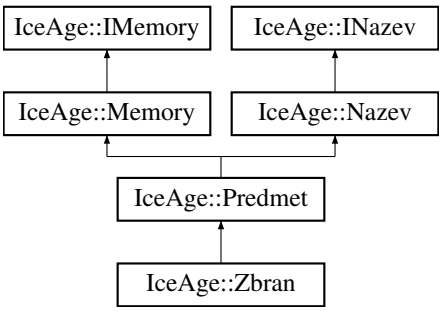
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/ZapadniPlanina.h
- cpp/ZapadniPlanina.cpp

6.38 Dokumentace třídy IceAge::Zbran

`#include <Zbran.h>`

Diagram dědičnosti pro třídu IceAge::Zbran



Veřejné metody

- short getPoskozeni ()

  Metoda vracejici hodnotu poskozeni.
- short getAtribut ()

  Podedena metoda vracejici hodnotu poskozeni.
- Zbran (std::string novyNazev, short novePoskozeni)

  Konstruktor.
- ~Zbran ()

  Destruktor.

Privátní metody

- void setPoskozeni (short novePoskozeni)

  Metoda nastavujici hodnotu poskozeni.

Privátní atributy

- short m_poskozeni

  Atribut obsahujici hodnotu poskozeni, ktere muze tato zbran zpusobit.

Další zděděné členy

6.38.1 Detailní popis

Trida, ktera znazorunuje predmet typu Zbran a implementuje pro nej rozhrani.

Definice je uvedena na řádku 14 v souboru Zbran.h.

6.38.2 Dokumentace konstruktoru a destruktoru

6.38.2.1 IceAge::Zbran::Zbran (std::string *novyNazev,* short *novePoskozeni*)

Konstruktor.

Konstruktor ve kterem se nastavuje poskozeni a typ predmetu se nastavuje na 'w'.

Definice je uvedena na řádku 24 v souboru Zbran.cpp.

6.38.2.2 IceAge::Zbran::~Zbran ( )

Destruktor.

Destruktor

Definice je uvedena na řádku 32 v souboru Zbran.cpp.

6.38.3 Dokumentace k metodám

6.38.3.1 short IceAge::Zbran::getAtribut ( ) [virtual]

Podedena metoda vracejici hodnotu poskozeni.

Podedena metoda vracejici hodnotu poskozeni.

Implementuje IceAge::Predmet.

Definice je uvedena na řádku 38 v souboru Zbran.cpp.

6.38.3.2 short IceAge::Zbran::getPoskozeni ( )

Metoda vracejici hodnotu poskozeni.

Metoda vracejici hodnotu poskozeni.

Definice je uvedena na řádku 9 v souboru Zbran.cpp.

6.38.3.3 void IceAge::Zbran::setPoskozeni (short *novePoskozeni*) [private]

Metoda nastavujici hodnotu poskozeni.

Metoda nastavujici hodnotu poskozeni. Hodnota poskozeni muze byt jen nezaporna.

Definice je uvedena na řádku 16 v souboru Zbran.cpp.

6.38.4 Dokumentace k datovým členům

6.38.4.1 short IceAge::Zbran::m_poskozeni [private]

Atribut obsahujici hodnotu poskozeni, ktere muze tato zbran zpusobit.

Definice je uvedena na řádku 17 v souboru Zbran.h.

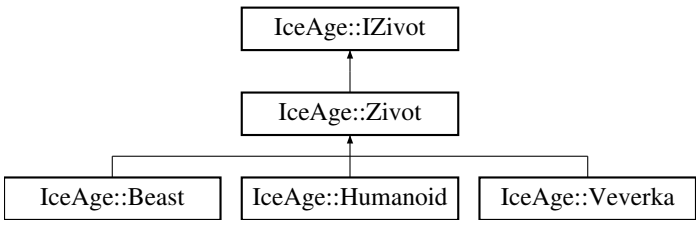
Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/Zbran.h
- cpp/Zbran.cpp

6.39 Dokumentace třídy IceAge::Zivot

`#include <Zivot.h>`

Diagram dědičnosti pro třídu IceAge::Zivot



Veřejné metody

- void changeZivotAktualni (short hodnotaZmeny)
  Metoda menici hodnotu aktualnihoZivota.
- short getZivotAktualni ()
  Metoda vracejici hodnotu aktualnihoZivota.
- short getZivotMaximalni ()
  Metoda vracejici hodnotu maximalnihoZivota.
- Zivot (short novyZivot, short novyMaximalniZivot)
  Konstruktor.
- virtual ~Zivot ()
  Virtualni Destruktor.
- virtual short utok ()=0
  Virtualni metoda vracejici hodnotu celkoveho Utoku.
- virtual short obrana ()=0
  Virtualni metoda vracejici hodnotu celkove Obrany.

Chráněné metody

- void setZivotAktualni (short novyZivotAktualni)
  Metoda nastavujici zivotAktualni.
- void setZivotMaximalni (short novyZivotMaximalni)
  Metoda nastavujici zivotMaximalni.

Chráněné atributy

- short m_zivotAktualni

  Aktualni hodnota zivota.
- short m_zivotMaximalni

  Maximalni hodnota zivota.

6.39.1 Detailní popis

Abstratkni trida, ktera implementuje rozhrani vsech dedenych trid, tykajicich se zivota.

Definice je uvedena na řádku 14 v souboru Zivot.h.

6.39.2 Dokumentace konstruktoru a destruktoru

6.39.2.1 IceAge::Zivot::Zivot (short *novyZivot,* short *novyMaximalniZivot*)

Konstruktor.

Konstruktor, ve kterem se nastavuje zivot. Nejdrive musi byt metoda setZivotMazimalni(). V opacnem pripade se do aktualniZivot nastavi nahodna hodnota.

Definice je uvedena na řádku 62 v souboru Zivot.cpp.

6.39.2.2 IceAge::Zivot::~Zivot ( ) [virtual]

Virtualni Destruktor.

Destruktor

Definice je uvedena na řádku 71 v souboru Zivot.cpp.

6.39.3 Dokumentace k metodám

6.39.3.1 void IceAge::Zivot::changeZivotAktualni (short *hodnotaZmeny*) [virtual]

Metoda menici hodnotu aktualnihoZivota.

Metoda menici zivotAktualni. V pripade, ze zdravi bude mensi nebo rovno nule, hra se ukonci a uvolni se pamet. Toto opatreni se tyka stylu Doby ledove, Scrat nezabiji a sam nesmi byt zabit. Zde se provadi kontrola, zda jeste zije.

Implementuje IceAge::IZivot.

Definice je uvedena na řádku 10 v souboru Zivot.cpp.

6.39.3.2 short IceAge::Zivot::getZivotAktualni ( ) [virtual]

Metoda vracejici hodnotu aktualnihoZivota.

Metoda vracejici hodnotu aktualnihoZivota.

Implementuje IceAge::IZivot.

Definice je uvedena na řádku 27 v souboru Zivot.cpp.

6.39.3.3 short IceAge::Zivot::getZivotMaximalni ( ) [virtual]

Metoda vracejici hodnotu maximalnihoZivota.

Metoda vracejici hodnotu maximalnihoZivota.

Implementuje IceAge::IZivot.

Definice je uvedena na řádku 34 v souboru Zivot.cpp.

6.39.3.4 virtual short IceAge::Zivot::obrana ( ) [pure virtual]

Virtualni metoda vracejici hodnotu celkove Obrany.

Implementováno v IceAge::Veverka, IceAge::Humanoid a IceAge::Beast.

6.39.3.5 void IceAge::Zivot::setZivotAktualni (short *novyZivotAktualni*) [protected], [virtual]

Metoda nastavujici zivotAktualni.

Metoda nastavujici zivotAktualni. Tato hodnotu musi byt kladna. V opacnem pripade je zivot nastaven na 1; Pokud nova hodnota presahne velikost maximalniho zivota, je nastavena maximalni povolena hodnota.

Implementuje IceAge::IZivot.

Definice je uvedena na řádku 42 v souboru Zivot.cpp.

6.39.3.6 void IceAge::Zivot::setZivotMaximalni (short *novyZivotMaximalni*) [protected], [virtual]

Metoda nastavujici zivotMaximalni.

Metoda nastavujici zivotMaximalni. Pokud je nova hodnota mensi jak 0, je nastaven na 5.

Implementuje IceAge::IZivot.

Definice je uvedena na řádku 54 v souboru Zivot.cpp.

6.39.3.7 virtual short IceAge::Zivot::utok ( ) [pure virtual]

Virtualni metoda vracejici hodnotu celkoveho Utoku.

Implementováno v IceAge::Veverka, IceAge::Humanoid a IceAge::Beast.

6.39.4 Dokumentace k datovým členům

6.39.4.1 short IceAge::Zivot::m_zivotAktualni [protected]

Aktualni hodnota zivota.

Definice je uvedena na řádku 17 v souboru Zivot.h.

6.39.4.2 short IceAge::Zivot::m_zivotMaximalni [protected]

Maximalni hodnota zivota.

Definice je uvedena na řádku 18 v souboru Zivot.h.

Dokumentace pro tuto třídu byla generována z následujících souborů:

- hlavicky/Zivot.h
- cpp/Zivot.cpp

Chapter 7

Dokumentace souborů

7.1 Dokumentace souboru cpp/Beast.cpp

```
#include "../hlavicky/Beast.h"
```

7.2 Beast.cpp

```
00001
00004 #include "../hlavicky/Beast.h"
00005
00009 short IceAge::Beast::getZurivost() {
00010     return this->m_zurivost;
00011 }
00012
00016 void IceAge::Beast::setZurivost(short novaZurivost) {
00017     this->m_zurivost=novaZurivost;
00018 }
00019
00023 short IceAge::Beast::utok() {
00024     return this->getSila() + this->getZurivost();
00025 }
00026
00030 short IceAge::Beast::obrana() {
00031     return this->getOdolnost();
00032 }
00033
00037 IceAge::Beast::Beast(short novaZurivost, std::string novyNazev, short novaSila,
      std::string novyPopis, short novaOdolnost, short novyZivot)
00038                                     :Memory(), Zivot(novyZivot,novyZivot),
      Nazev(novyNazev), Popis(novyPopis), Staty(novaSila,novaOdolnost){
00039     this->setZurivost(novaZurivost);
00040 }
00041
00045 IceAge::Beast::~Beast(){
00046 }
```

7.3 Dokumentace souboru cpp/BuilderBytost.cpp

```
#include "../hlavicky/BuilderBytost.h"
```

7.4 BuilderBytost.cpp

```
00001
00004 #include "../hlavicky/BuilderBytost.h"
00005
00009 IceAge::BuilderBytost::BuilderBytost(
      IceAge::BuilderPredmet* novyBuilder) {
00010     this->setBuilder(novyBuilder);
00011     this->m_beast=nullptr;
00012     this->m_humanoid=nullptr;
00013 }
00014
00015
00019 void IceAge::BuilderBytost::setBuilder(
      IceAge::BuilderPredmet* novyBuilder) {
00020     this->m_builder=novyBuilder;
00021 }
00022
00026 IceAge::Predmet* IceAge::BuilderBytost::createPredmet(
      std::string novyNazev, short novaHodnota) {
00027     this->m_builder->createPredmet(novyNazev, novaHodnota);
00028     return this->m_builder->getPredmet();
00029 }
00030
00034 IceAge::Beast* IceAge::BuilderBytost::getBeast() {
00035     IceAge::Beast*third=this->m_beast;
00036     this->m_beast=nullptr;
00037     return third;
00038 }
00039
00043 IceAge::Humanoid* IceAge::BuilderBytost::getHumanoid() {
00044     IceAge::Humanoid*third=this->m_humanoid;
00045     this->m_humanoid=nullptr;
00046     return third;
00047 }
00048
00052 void IceAge::BuilderBytost::createBeast(short novaZurivost, std::string
      novyNazev, short novaSila, std::string novyPopis, short novaOdolnost, short novyZivot) {
00053     this->m_beast=new IceAge::Beast(novaZurivost, novyNazev, novaSila, novyPopis,
      novaOdolnost, novyZivot);
00054 }
00055
00056 void IceAge::BuilderBytost::createHumanoid(short novyZivot,
      std::string novyPopis, short novaSila, short novaOdolnost,
00057                                            std::string novyNazevPredmetu1, std::string novyNazevPredmetu2,
       std::string novyNazevPredmetu3,
00058                                            short novaHodnotaPredmetu1, short novaHodnotaPredmetu2, short
      novaHodnotaPredmetu3) {
00059
00060     this->m_humanoid=new IceAge::Humanoid(novyZivot, "Ruzovy dvounozec",
      novyPopis, novaSila, novaOdolnost,
00061                                           this->createPredmet(novyNazevPredmetu1,
      novaHodnotaPredmetu1),
00062                                           this->createPredmet(novyNazevPredmetu2,
      novaHodnotaPredmetu2),
00063                                           this->createPredmet(novyNazevPredmetu3,
      novaHodnotaPredmetu3));
00064 }
00065
00069 IceAge::BuilderBytost::~BuilderBytost(){
00070     if(this->m_beast != nullptr) delete this->m_beast;
00071     if(this->m_humanoid != nullptr) delete this->m_humanoid;
00072     if(this->m_builder != nullptr) delete this->m_builder;
00073 }
```

7.5 Dokumentace souboru cpp/BuilderLokace.cpp

```
#include "../hlavicky/BuilderLokace.h"
```

7.6 BuilderLokace.cpp

```
00001
```

```
00004 #include "../hlavicky/BuilderLokace.h"
00005
00009 IceAge::BuilderLokace::BuilderLokace(
      IceAge::BuilderBytost* novyBuilder) {
00010     this->setBuilder(novyBuilder);
00011     this->m_lokace=nullptr;
00012 }
00013
00017 void IceAge::BuilderLokace::setBuilder(
      IceAge::BuilderBytost* novyBuilder) {
00018     this->m_builderBytost=novyBuilder;
00019 }
00020
00024 IceAge::Lokace* IceAge::BuilderLokace::getLokace() {
00025     return this->m_lokace;
00026 }
00027
00031 IceAge::Humanoid* IceAge::BuilderLokace::createHumanoid
      (short novyZivot, std::string novyPopis, short novaSila, short novaOdolnost,
00032                                                 std::string novyNazevPredmetu1, std::string
      novyNazevPredmetu2, std::string novyNazevPredmetu3,
00033                                                 short novaHodnotaPredmetu1, short novaHodnotaPredmetu2
      , short novaHodnotaPredmetu3) {
00034     this->m_builderBytost->createHumanoid(novyZivot, novyPopis, novaSila,
      novaOdolnost, novyNazevPredmetu1, novyNazevPredmetu2, novyNazevPredmetu3,
00035                                           novaHodnotaPredmetu1, novaHodnotaPredmetu2,
      novaHodnotaPredmetu3);
00036     return this->m_builderBytost->getHumanoid();
00037 }
00038
00042 IceAge::Beast* IceAge::BuilderLokace::createBeast(short
      novaZurivost, std::string novyNazev, short novaSila, std::string novyPopis, short novaOdolnost, short novyZivot)
       {
00043     this->m_builderBytost->createBeast( novaZurivost, novyNazev, novaSila,
      novyPopis, novaOdolnost, novyZivot);
00044     return this->m_builderBytost->getBeast();
00045 }
00046
00050 void IceAge::BuilderLokace::createLokace() {
00051
00052     std::ifstream seznam, popisLokace;
00053     seznam.open("Lokace/.seznam");
00054     if(!seznam.is_open())
00055     {
00056         std::cerr << "Chyby seznam lokaci. Program ukoncen.";
00057         IceAge::MemoryArbiter::cycleDelete();
00058         exit(-1);
00059     }
00060 //    std::cout << "Soubor s lokacemi otevren" << std::endl;
00061
00062     std::string radek, nazevL, popisL, popisLCast, rekaL;
00064 //    IceAge::Lokace *pridavanaLokace;
00065
00066     while(getline(seznam, radek))
00067     {
00068 //        std::cout << "Otevira se " << radek << std::endl;
00069         popisLokace.open(radek);
00070         if(!popisLokace.is_open())
00071         {
00072             std::cerr << "Chybi soubor: " << radek << std::endl;
00073             IceAge::MemoryArbiter::cycleDelete();
00074             exit(-1);
00075         }
00077         getline(popisLokace, nazevL);           //************************
00078         getline(popisLokace, rekaL);            //                        **
00079         popisL=""; popisLCast="";               //nacitani popisu         **
00080         do                                      //                        **
00081         {                                       //                        **
00082             popisL+= popisLCast;                //                        **
00083             getline(popisLokace, popisLCast);   //                        **
00084         }while(popisLCast != "***");            //************************
00085
00086
00087         this->m_lokace=new IceAge::Lokace(nazevL, popisL);
00089         if(rekaL == "1") this->m_lokace->pridejReku();
00091 //nacteni poctu Humanoidu
00092         this->nactiHumanoidy(popisLokace);
00093         this->nactiBeasty(popisLokace);
00095         while(!popisLokace.fail())
00096         {
00097             //std::cout << "jsem tu" << std::endl;
00098             getline(popisLokace, radek);
00099                                 //std::cout << "Nacetl jsem " << radek << std::endl;
00100             for(unsigned int i=0; i<this->m_vektorLokaci.size(); i++)
00101             {
00102                 //std::cout << radek;
00103                 if(radek == (this->m_vektorLokaci.at(i)->getNazevLokace()))
```

```
00104                 {
00105                                     //std::cout << radek + " " + this->m_vektorLokaci.at(i)->getNazevLokace()
      << std::endl;
00106                     this->m_lokace->pridejLokaci(this->
     m_vektorLokaci.at(i));
00107                                     // std::cout << this->m_lokace->getNazevLokace() << " dostava " <<
      this->m_vektorLokaci.at(i)->getNazevLokace() << std::endl;
00108                     this->m_vektorLokaci.at(i)->pridejLokaci(this->
     m_lokace);
00109                                     //std::cout << this->m_vektorLokaci.at(i)->getNazevLokace() << " dostava "
      << this->m_lokace->getNazevLokace() << std::endl;
00110                 }
00111             }
00112         }
00113
00114 //        this->m_vektorLokaci.push_back(pridavanaLokace);
00115         this->m_lokace->setPrikaz();
00116         this->m_vektorLokaci.push_back(this->m_lokace);
00117         this->m_lokace=nullptr;
00118         popisLokace.close();
00119     }
00120
00121
00122     seznam.close();
00124     this->m_lokace=this->m_vektorLokaci.at(0);
00126     this->m_vektorLokaci.clear();
00127 }
00128
00132 void IceAge::BuilderLokace::nactiHumanoidy(std::ifstream &popisLokace)
     {
00133     short pocet=0;
00134     std::string radek="";
00135     popisLokace >> pocet;           popisLokace.ignore();
00137 //    std::cout << pocet << std::endl;
00138     if(pocet <=0)
00139     {
00140 //        std::cout << "Nejsou tu lidi" << std::endl;
00141         getline(popisLokace,radek);
00142         return;
00143     }
00144     std::string popis, nazevPredmetu1, nazevPredmetu2, nazevPredmetu3;
00145     short zivot, sila, odolnost, hodnotaPredmetu1, hodnotaPredmetu2, hodnotaPredmetu3;
00146
00147     for(short i=0; i<pocet; i++)
00148     {
00149         getline(popisLokace,popis);
00150         popisLokace >> sila;                        popisLokace.ignore();
00151         popisLokace >> odolnost;                    popisLokace.ignore();
00152         popisLokace >> zivot;                       popisLokace.ignore();
00153         getline(popisLokace,nazevPredmetu1);
00154         popisLokace >> hodnotaPredmetu1;            popisLokace.ignore();
00155         getline(popisLokace,nazevPredmetu2);
00156         popisLokace >> hodnotaPredmetu2;            popisLokace.ignore();
00157         getline(popisLokace,nazevPredmetu3);
00158         popisLokace >> hodnotaPredmetu3;            popisLokace.ignore();
00159
00160         this->m_lokace->pridejHumanoid(this->
     createHumanoid(zivot,popis,sila,odolnost,
00161                                                     nazevPredmetu1,nazevPredmetu2,nazevPredmetu3,
00162                                                     hodnotaPredmetu1, hodnotaPredmetu2, hodnotaPredmetu3));
00163
00164     }
00165
00166     getline(popisLokace,radek);
00168 }
00169
00173 void IceAge::BuilderLokace::nactiBeasty(std::ifstream &popisLokace){
00174     short pocet=0;
00175     std::string radek="";
00176     popisLokace >> pocet; popisLokace.ignore();
00178     if(pocet <=0)
00179     {
00180         getline(popisLokace,radek);
00181         return;
00182     }
00183 //    std::cout << "jsem tady" << std::endl;
00184     std::string popis, nazev;
00185     short zurivost, zivot, sila, odolnost;
00186
00187     for(short i=0; i<pocet; i++)
00188     {
00189         getline(popisLokace,nazev);
00190         getline(popisLokace,popis);
00191         popisLokace >> zurivost;            popisLokace.ignore();
00192         popisLokace >> sila;                popisLokace.ignore();
00193         popisLokace >> odolnost;            popisLokace.ignore();
00194         popisLokace >> zivot;               popisLokace.ignore();
```

```
00195
00196         this->m_lokace->pridejBeast(this->createBeast(zurivost,nazev,sila,
      popis,odolnost,zivot));
00197
00198     }
00199
00200     getline(popisLokace,radek);
00201 //    std::cout << radek;
00202 }
00203
00207 IceAge::BuilderLokace::~BuilderLokace(){
00208     if(this->m_builderBytost != nullptr) delete this->
      m_builderBytost;
00209     if(this->m_lokace != nullptr) delete this->m_lokace;
00210
00211     for(unsigned int i=0; i<m_vektorLokaci.size(); i++)
00212         if(this->m_vektorLokaci.at(i) != nullptr) delete this->
      m_vektorLokaci.at(i);
00213 }
```

7.7 Dokumentace souboru cpp/BuilderPredmet.cpp

```
#include "../hlavicky/BuilderPredmet.h"
```

7.8 BuilderPredmet.cpp

```
00001
00004 #include "../hlavicky/BuilderPredmet.h"
00005
00009 void IceAge::BuilderPredmet::createPredmet(std::string novyNazev,
      short novaHodnota) {
00010     this->m_pripravovany=new IceAge::Zbran(novyNazev,novaHodnota);
00011 }
00012
00016 IceAge::Predmet* IceAge::BuilderPredmet::getPredmet() {
00017     IceAge::Predmet*third=this->m_pripravovany;
00018     this->m_pripravovany=nullptr;
00019     return third;
00020 }
00021
00025 IceAge::BuilderPredmet::BuilderPredmet() {
00026     this->m_pripravovany=nullptr;
00027 }
00028
00032 IceAge::BuilderPredmet::~BuilderPredmet(){
00033     if(this->m_pripravovany != nullptr) delete this->
      m_pripravovany;
00034 }
```

7.9 Dokumentace souboru cpp/Command.cpp

```
#include "../hlavicky/Command.h"
```

7.10 Command.cpp

```
00001
00004 #include "../hlavicky/Command.h"
00005
00009 IceAge::Command::Command(std::string novyNazev) : Nazev(novyNazev) {
00010 }
00011
00015 IceAge::Command::~Command() {
00016 }
```

7.11 Dokumentace souboru cpp/EngineIceAge.cpp

```
#include "../hlavicky/EngineIceAge.h"
```

7.12 EngineIceAge.cpp

```
00001
00004 #include "../hlavicky/EngineIceAge.h"
00005
00006 IceAge::EngineIceAge*
      IceAge::EngineIceAge::s_vlastniInstance=nullptr;
00011 /*static*/ IceAge::EngineIceAge*
      IceAge::EngineIceAge::getInstance() {
00012     if(s_vlastniInstance == nullptr) s_vlastniInstance=new
      IceAge::EngineIceAge(new IceAge::LocationDirector
00013                                                                    (new
      IceAge::BuilderLokace
00014                                                                    (new
      IceAge::BuilderBytost
00015                                                                    (new
      IceAge::BuilderPredmet())));
00016     return s_vlastniInstance;
00017 }
00018
00022 IceAge::EngineIceAge::EngineIceAge(
      IceAge::LocationDirector* novyDirector) {
00023     this->setDirector(novyDirector);
00024     this->m_lokace=nullptr;
00025     this->m_veverka=nullptr;
00026 }
00027
00031 IceAge::Veverka* IceAge::EngineIceAge::getVeverka() {
00032     return this->m_veverka;
00033 }
00034
00038 void IceAge::EngineIceAge::createVeverka() {
00039     this->m_veverka=IceAge::Veverka::getInstance();
00040 }
00041
00045 void IceAge::EngineIceAge::startHry() {
00046     if(!this->inicializaceLokaci())
00047     {
00048         return;
00049     }
00050     this->createVeverka();
00051
00052     IceAge::Hra *PJ = new IceAge::Hra(this->m_lokace);
00053
00054     PJ->zacniHrat(this->m_veverka);
00055
00056     delete PJ;
00057 }
00058
00062 bool IceAge::EngineIceAge::inicializaceLokaci() {
00063     this->m_lokace=this->m_directorLokaci->createLokace();
00064 //    std::cout<<"lokace ok"<<std::endl;
00065     if(this->m_lokace != nullptr) return true;
00066     else return false;
00067 }
00071 void IceAge::EngineIceAge::setDirector(
      IceAge::LocationDirector* novyDirector){
00072     this->m_directorLokaci=novyDirector;
00073 }
00074
00078 IceAge::EngineIceAge::~EngineIceAge(){
00079     if(this->m_directorLokaci != nullptr) delete this->
      m_directorLokaci;
00080     if(this->m_veverka != nullptr) delete this->m_veverka;
00081     if(this->m_lokace != nullptr) delete this->m_lokace;
00082 }
```

7.13 Dokumentace souboru cpp/Hra.cpp

```
#include "../hlavicky/Hra.h"
```

7.14 Hra.cpp

```
00001
00004 #include "../hlavicky/Hra.h"
00005
00009 IceAge::Hra::Hra(IceAge::Lokace* novaAktualniLokace) {
00010     this->setAktualniLokace(novaAktualniLokace);
00011     this->m_obrazek=0;
00012 }
00013
00017 void IceAge::Hra::zmenLokaci(IceAge::Veverka* scrat) {
00018     char volba= 'a';
00019     unsigned int volbaCislo=0;
00020     std::cout << std::endl;
00021     this->m_aktualniLokace->vypisLokace();
00022 //    std::cout << std::endl;
00023
00024     while(true){
00025         std::cout << "Vyber akci:";
00026         std::cin >> volba;std::cin.ignore(); volbaCislo=volba - '0';
00027         std::cout << std::endl;
00028         if(volbaCislo < (this->m_aktualniLokace->
      getVelikostVektoruLokaci())){
00029             this->zmenLokaci(volbaCislo, scrat);
00030             break;
00031         }
00032     }
00033
00034 //    std::cout << "[" << index << "] " <<std::endl; index++;
00035 }
00036
00040 void IceAge::Hra::zmenLokaci(unsigned int index,
      IceAge::Veverka* scrat) {
00041     if((this->m_aktualniLokace->getLokace(index)->
      getNazevLokace() == "Severni les") and (this->m_aktualniLokace->
      getNazevLokace() == "Jizni les")){
00043         if(scrat->getPravaRuka() == nullptr || scrat->getLevaRuka()== nullptr){
00044             std::cout << "Nemas se jak dostat pres reku." << std::endl;
00045         }
00046         else{
00047             std::cout << "Pouzivas svoji zbran jako padlo a plujes po lodce na druhou stranu" << std::endl;
00048             this->setAktualniLokace(this->m_aktualniLokace->
      getLokace(index));
00049         }
00050     }
00051     else if((this->m_aktualniLokace->getLokace(index)->
      getNazevLokace() == "Jizni les") and (this->m_aktualniLokace->
      getNazevLokace() == "Severni les")){
00052         if(scrat->getPravaRuka() == nullptr || scrat->getLevaRuka()== nullptr){
00053             std::cout << "Nemas se jak dostat pres reku." << std::endl;
00054         }
00055         else{
00056             std::cout << "Pouzivas svoji zbran jako padlo a plujes po lodce na druhou stranu" << std::endl;
00057             this->setAktualniLokace(this->m_aktualniLokace->
      getLokace(index));
00058         }
00059     }
00060     else this->setAktualniLokace(this->m_aktualniLokace->
      getLokace(index));
00061 }
00062
00066 void IceAge::Hra::setAktualniLokace(IceAge::Lokace*
      novaAktualniLokace){
00067     this->m_aktualniLokace=novaAktualniLokace;
00068 }
00069
00073 void IceAge::Hra::vypisAktualniLokaci(){
00074     std::cout << std::endl;
00075     std::cout << "Nazev mista: " << this->m_aktualniLokace->
      getNazevLokace() << std::endl;
00076     std::cout << "Popis: " << this->m_aktualniLokace->
      getPopisLokace() << std::endl;
00077     std::cout << std::endl;
00078 }
00079
00083 void IceAge::Hra::vypisAsciObrazek(){
00084         this->m_obrazek++;
00085         switch (this->m_obrazek){
00086         case 1:
00087 std::cout<<"  _,                          _"                 << std::endl;
00088 std::cout<<".'  `.                  ___.>\"''-..-."          << std::endl;
00089 std::cout<<"`-.   ;           .--\"\"\"\"         .-._@;"      << std::endl;
00090 std::cout<<"   ;  !_.--..._ .'      /     .[_@'`'."          << std::endl;
00091 std::cout<<"  ;            /       : .'  ; :_..._  `."       << std::endl;
00092 std::cout<<"  :           ;        ;[   _T-\"  `.'-. `-."    << std::endl;
00093 std::cout<<"   \\        .-:      ; `.`-=_,88p.   _.}.-\""   << std::endl;
```

```
00094 std::cout<<"    `-.__.-'    \\    /L._ Y\",P$T888;  \"\""    << std::endl;
00095 std::cout<<"              .-'_.-'  / ;$$$$$$]8P;"           << std::endl;
00096 std::cout<<"             \\ /     / / \"Y$$P\" ^\""          << std::endl;
00097 std::cout<<"              ;\\_    `.\\_._"                   << std::endl;
00098 std::cout<<"              ]__\\     \\___;"                  << std::endl;
00099
00100 std::cin.ignore();
00101 std::cout<<std::endl;
00102 std::cout<<std::endl;
00103 std::cout<<std::endl;
00104
00105
00106
00107             break;
00108         case 2:
00109 std::cout<<"                ___"          << std::endl;
00110 std::cout<<"         .-\"\"\"\"   \"."         << std::endl;
00111 std::cout<<"        /         __'-."         << std::endl;
00112 std::cout<<"       ;       ..sssSSSS;"       << std::endl;
00113 std::cout<<"       ;   ;          ;"         << std::endl;
00114 std::cout<<"        '.'  ..sssSSSSSS;"       << std::endl;
00115 std::cout<<"        ;        \"\"\"\"\"\"\";"         << std::endl;
00116 std::cout<<"       ;    ...sssssSSSSS;"      << std::endl;
00117 std::cout<<"       ;          \"\"\"\"\"\";"        << std::endl;
00118 std::cout<<"       ;              ;"         << std::endl;
00119 std::cout<<"       ;     ....sssSS/"         << std::endl;
00120 std::cout<<"       ;          \"\"\"/"         << std::endl;
00121 std::cout<<"        ;          .'"          << std::endl;
00122 std::cout<<"        ;    .-\"\"\"\"-."         << std::endl;
00123 std::cout<<"         '-.'  _..ssS,"         << std::endl;
00124 std::cout<<"         .'  \"\"  _..sSs"         << std::endl;
00125 std::cout<<"        /__    \"\"  _.sSS."         << std::endl;
00126 std::cout<<"      .-\"\" `-.    ___     ; _"         << std::endl;
00127 std::cout<<"     /_..gg$$$pp'___`.   .' `>."         << std::endl;
00128 std::cout<<"   ,s$$$$$$$$B;\"    `;\"\";  .' ;"         << std::endl;
00129 std::cout<<" :$$$$$$$$$P\"`._():   `-`_O.'"         << std::endl;
00130 std::cout<<":$$$$$$$$P          '   `-."         << std::endl;
00131 std::cout<<"$$$$$$$$\"    _,,-.       :     ;"         << std::endl;
00132 std::cout<<"$$$$$$$!b._g$$$$$-.     ;     `."         << std::endl;
00133 std::cout<<":$$$$$$$$$$$$$$$$$P j\\    :_._/"         << std::endl;
00134 std::cout<<" T$$$$$$$$$$$$$$$P  | :    ;"         << std::endl;
00135 std::cout<<"  \"T$$$$$$$$$$$P\";  ;_;    :"         << std::endl;
00136 std::cout<<"    \"^T$$$$$P^\"; : //:   __!"         << std::endl;
00137 std::cout<<"        | |     : ; `.: .mMMM:"         << std::endl;
00138 std::cout<<"        ) :_    ) '-.   'MMMP'"         << std::endl;
00139 std::cout<<"       `.i_;I  '-._i.'"         << std::endl;
00140
00141 std::cin.ignore();
00142 std::cout<<std::endl;
00143 std::cout<<std::endl;
00144 std::cout<<std::endl;
00145             break;
00146         case 3:
00147 std::cout<<"                              ,"         << std::endl;
00148 std::cout<<"         ,::.:::::.          / `."         << std::endl;
00149 std::cout<<"       ;::        `.        (  ,--."         << std::endl;
00150 std::cout<<"       ,;           :   _,--'\"\\/\"\"-.\\_"         << std::endl;
00151 std::cout<<"      ,:            `./ ,---./( O )) ';"         << std::endl;
00152 std::cout<<"      ;  `.          _,'    (  `-' ) /_."         << std::endl;
00153 std::cout<<"      ;   :        ,'        \\     , (o\\\\"         << std::endl;
00154 std::cout<<"       ;  :        \\  \\-.  -.__--'   \\/' )"         << std::endl;
00155 std::cout<<"       ;  ;        /\\ (    `-._`-._   \\/"         << std::endl;
00156 std::cout<<"        ';,       ; : |      -.`._'\\   `."         << std::endl;
00157 std::cout<<"          ;      ;  : `-.,-..  )  \\\'\\   ^."         << std::endl;
00158 std::cout<<"           ;     ;   `.__   )))\\\ ) ('.\\    \\"         << std::endl;
00159 std::cout<<"            ;   ;       `-'///(, \\ \\ \\\)  ,ooo."         << std::endl;
00160 std::cout<<"             ;,;      ;     `` ))))_;'(  88888p"         << std::endl;
00161 std::cout<<"              ;       ;        ((-='--',-,Y8888'"         << std::endl;
00162 std::cout<<"              ;      :        ;     ,:'-'  '\"'"         << std::endl;
00163 std::cout<<"               ;     `        ;      |"         << std::endl;
00164 std::cout<<"                ;      (_  __   _,-'"         << std::endl;
00165 std::cout<<"                 `---.   ;\"(,-' /                                  ____"         << std::endl;
00166 std::cout<<"                      \\ (__/\\\\_`-.-.                          _____/"         << std::endl;
00167 std::cout<<"                    ,(( '/\\/\\/\\`-;;))                 ______//"         << std::endl;
00168 std::cout<<"                   ((\\'' /\\/\\/\\/\\/\\/'/\\      _____/  ____/"         << std::endl;
00169 std::cout<<"                   /' /\\/\\/\\/\\/\\/\\/\\/\\)  __/  _____/"         << std::endl;
00170 std::cout<<"                  (\\/\\/\\/\\/\\/\\/\\/\\/\\/\\_/ _/ ___/"         << std::endl;
00171 std::cout<<"                   `-|\"\"--\"-.___,--'|-'__/"         << std::endl;
00172 std::cout<<"                     |              | /"         << std::endl;
00173 std::cout<<"                     |          __,--'"         << std::endl;
00174 std::cout<<"                     _\\,----\"\"\"'"         << std::endl;
00175
00176 std::cin.ignore();
00177 std::cout<<std::endl;
00178 std::cout<<std::endl;
00179 std::cout<<std::endl;
00180             break;
```

```
00181         case 4:
00182              break;
00183         case 5:
00184              break;
00185         case 6:
00186              break;
00187         case 7:
00188             break;
00189         default:
00190             break;
00191         }
00192 }
00193
00194 void IceAge::Hra::intro(IceAge::Veverka *scrat){
00195     char volba='a';
00196     bool cyklus=true,cyklus2=true;
00197     short index=0;
00199     std::cout << "Bylo nebylo.." << std::endl;
00200     std::cout << "Za davnych casu zila jedna veverka." << std::endl;
00201     this->vypisAsciObrazek();
00203     std::cout << "Hledala misto, kam by si schovala svuj orisek." << std::endl;
00204     this->vypisAsciObrazek();
00206     std::cout << "[0] Hledej dal" <<std::endl;
00207     std::cout << "Vyber akci:";
00208     std::cin >> volba;std::cin.ignore();
00209     while(cyklus)
00210     {
00211         switch(volba)
00212         {
00213             case '0':
00214                 cyklus=false;
00215                 break;
00216             default:
00217                 std::cout << "Vyber akci:";
00218                 std::cin >> volba;std::cin.ignore();
00219                 break;
00220         }
00221     }
00222     cyklus=true;
00223     while(cyklus)
00224     {
00225         cyklus2=true;
00226         std::cout << std::endl << "Nasel si ho!" << std::endl; std::cin.ignore();
00227         while(cyklus2)
00228         {
00229             std::cout << "[" << index << "] Hledej dal" <<std::endl; index++;
00230             std::cout << "[" << index << "] Vloz orisek" <<std::endl; index++;
00231             std::cout << "[" << index << "] Kde ze jsem?" <<std::endl; index=0;
00232             std::cout << "Vyber akci:";
00233             std::cin >> volba;std::cin.ignore();
00234             std::cout << std::endl;
00235             switch(volba)
00236             {
00237                 case '0':
00238                     cyklus2=false;
00239                     continue;
00240                     break;
00241                 case '1':
00242                     cyklus=false; cyklus2=false;
00243                     break;
00244                 case '2':
00245                     this->vypisAktualniLokaci();
00246                     continue;
00247                     break;
00248                 default:
00249                     break;
00250             }
00251         }
00252     }
00253     cyklus=true; cyklus2=true;
00255     this->vypisAsciObrazek();
00256     std::cout << "Povedlo se ti rozbit ledovec na kterem jsi byl!!" << std::endl;
00257     std::cout << "Ledovec se rozpada a sjizdis na lavine do udoli" << std::endl;
00258     std::cout << std::endl;
00259
00260     this->zmenLokaci(0,scrat);
00262     std::cout << "[" << index << "] Kde ze jsem?" <<std::endl; index=0;
00263     while(cyklus2)
00264     {
00265             std::cout << "Vyber akci:";
00266             std::cin >> volba;std::cin.ignore();
00267             switch(volba)
00268             {
00269                 case '0':
00270                     this->vypisAktualniLokaci();
00271                     cyklus2=false;
00272                     break;
```

```
00273                 default:
00274                     break;
00275             }
00276     }
00277     cyklus2=true;
00278     scrat->zahodPravaRuka();
00279 
00280     std::cout << "Opatrne se zvedas na nohy. Chybi ti orisek!!" << std::endl;
00281     std::cout << "Vidis velkeho Supomuta, jak ti s nim chce odletet!" << std::endl;
00282 
00284     std::cout << std::endl;
00285     std::cout << "[" << index << "] Zautocit" <<std::endl; index++;
00286     std::cout << "[" << index << "] Utect" <<std::endl; index++;
00287     std::cout << "[" << index << "] Schovat se" <<std::endl; index=0;
00288     while(cyklus2)
00289     {
00290             std::cout << "Vyber akci:";
00291             std::cin >> volba;std::cin.ignore();
00292             switch(volba)
00293             {
00294                 case '0':
00295                     boj(scrat, this->m_aktualniLokace->
      getBeast(3));
00296                     std::cout << "Dostal si ranu paratem. Supomut ti odlita pred nosem." << std::endl;
00297                     cyklus2=false;
00298                     break;
00299                 case '1':
00300                     std::cout << "Supomut odleta pryc i s tvym oriskem" << std::endl;
00301                     cyklus2=false;
00302                     break;
00303                 case '2':
00304                     std::cout << "Schovavas se za kousek ledu. Supomut si te nevsima a odleta s tvym
       oriskem" << std::endl;
00305                     cyklus2=false;
00306                     break;
00307                 default:
00308                     break;
00309             }
00310     }
00311     cyklus2=true;
00313     std::cout << std::endl;
00314     std::cout << "[" << index << "] Pronasleduj Supomuta" <<std::endl; index=0;
00315     while(cyklus2)
00316     {
00317             std::cout << "Vyber akci:";
00318             std::cin >> volba;std::cin.ignore();
00319             switch(volba)
00320             {
00321                 case '0':
00322                     std::cout << "Bezis jak nejrychleji muzes, ale cestu ti prekrizila reka." << std::endl;
00323                     std::cout << "Vidis jak se ti Supomut vzdaluje na obzoru." << std::endl;
00324                     this->zmenLokaci(1, scrat);
00325                     cyklus2=false;
00326                     break;
00327                 default:
00328                     break;
00329             }
00330     }
00331 }
00332 
00336 void IceAge::Hra::zacniHrat(IceAge::Veverka *scrat){
00337     char volba='a';
00338     bool cyklus=true,cyklus2=true;
00339     short index=0;
00341     this->intro(scrat);
00343     while(cyklus)
00344     {
00345         cyklus2=true;
00346 
00347         this->vypisAktualniLokaci();
00348 
00349         while(cyklus2)
00350         {
00351             std::cout << "[" << index << "] Jit jinam " <<std::endl; index++;
00352             std::cout << "[" << index << "] Kde ze jsem?" <<std::endl;index++;
00353             std::cout << "[" << index << "] Prozkoumat to tu" <<std::endl; index=0;
00354             std::cout << "Vyber akci:";
00355             std::cin >> volba;std::cin.ignore();
00356             switch(volba)
00357             {
00358                 case '0':
00359                     this->zmenLokaci(scrat);
00360                     cyklus2=false;
00361                     continue;
00362                     break;
00363                 case '1':
00364                     this->vypisAktualniLokaci();
```

```
00365                     continue;
00366                     break;
00367                 case '2':
00368                     this->m_aktualniLokace->prozkoumatLokaci(scrat);
00369                     break;
00370                 default:
00371                     break;
00372             }
00373         }
00374     }
00375 }
```

7.15 Dokumentace souboru cpp/Humanoid.cpp

```
#include "../hlavicky/Humanoid.h"
```

7.16 Humanoid.cpp

```
00001
00004 #include "../hlavicky/Humanoid.h"
00005
00009 IceAge::Humanoid::Humanoid(short novyZivot, std::string novyNazev, std::string
      novyPopis, short novaSila, short novaOdolnost, IceAge::Predmet* zbran1,
00010         IceAge::Predmet* zbran2, IceAge::Predmet* zbran3):
      IceAge::Inventar::Inventar(MAX_INVENTAR),
      IceAge::ObleceneVybaveniHumanoid(zbran1,zbran2,zbran3),
00011                                                     IceAge::Zivot::
      Zivot(novyZivot,novyZivot), IceAge::Nazev::Nazev(novyNazev),
00012                                                     IceAge::Popis::
      Popis(novyPopis), IceAge::Staty::Staty(novaSila,novaOdolnost) {
00013 }
00014
00018 short IceAge::Humanoid::utok() {
00019     short utokCelkem=0;
00020
00021     if(this->m_levaRuka != nullptr)    if(this->m_levaRuka->
      getTyp() == 'w') utokCelkem+=this->m_levaRuka->getAtribut();
00022     if(this->m_pravaRuka != nullptr)    if(this->m_pravaRuka->
      getTyp() == 'w') utokCelkem+=this->m_pravaRuka->getAtribut();
00023
00024     return this->getSila() + utokCelkem;
00025 }
00026
00030 short IceAge::Humanoid::obrana() {
00031 //    short obranaCelkem=0;
00032 /*
00033     if(this->m_levaRuka != nullptr)    if(this->m_levaRuka->getTyp() == 'c')
       obranaCelkem+=this->m_levaRuka->getAtribut();
00034     if(this->m_pravaRuka != nullptr)    if(this->m_pravaRuka->getTyp() == 'c')
       obranaCelkem+=this->m_pravaRuka->getAtribut();
00035 */
00036     return this->getOdolnost()/* + obranaCelkem*/;
00037 }
00038
00042 IceAge::Humanoid::~Humanoid(){
00043 }
```

7.17 Dokumentace souboru cpp/Inventar.cpp

```
#include "../hlavicky/Inventar.h"
```

7.18 Inventar.cpp

```
00001
00004 #include "../hlavicky/Inventar.h"
00005
00006
00010 unsigned short IceAge::Inventar::getMaximalniVelikost() {
00011     return this->m_maximalniVelikost;
00012 }
00013
00017 IceAge::Predmet* IceAge::Inventar::odeberPredmet() {
00018     std::cout << "Vyber predmet:" << std::endl;
00019     this->vypisObsah();
00020
00021     unsigned int x=this->volba();
00025     IceAge::Predmet *odeber=this->m_vektorPredmetu.at(x);
00026     this->m_vektorPredmetu.at(x)=nullptr;
00027     this->m_vektorPredmetu.pop_back();
00028     return odeber;
00029 }
00030
00034 void IceAge::Inventar::pridejPredmet(
      IceAge::Predmet* novyPredmet) {
00035     if(this->m_vektorPredmetu.size() >= this->
      getMaximalniVelikost()) std::cout << "Inventar je plny!" << std::endl;
00036     else this->m_vektorPredmetu.push_back(novyPredmet);
00037 }
00038
00042 void IceAge::Inventar::setMaximalniVelikost(unsigned short
      novaMaximalniVelikost) {
00043     if(novaMaximalniVelikost > 0)this->m_maximalniVelikost=novaMaximalniVelikost;
00044     else this->m_maximalniVelikost=1;
00045 }
00046
00051 void IceAge::Inventar::znicPredmet() {
00052     std::cout << "Vyber predmet k zahozeni:" << std::endl;
00053
00054     this->vypisObsah();
00055     unsigned int x = this->volba();
00056
00057     IceAge::Predmet *odeber=this->m_vektorPredmetu.at(x);
00058     this->m_vektorPredmetu.at(x)=this->m_vektorPredmetu.at(this->
      m_vektorPredmetu.size()-1);
00059     this->m_vektorPredmetu.at(this->m_vektorPredmetu.size()-1)=odeber;
00060     this->m_vektorPredmetu.pop_back();
00061 }
00062
00066 IceAge::Inventar::Inventar(unsigned short novaVelikost) {
00067     this->setMaximalniVelikost(novaVelikost);
00068 }
00069
00073 IceAge::Inventar::~Inventar() {
00074     for(unsigned int i=0; i<m_vektorPredmetu.size(); i++)
00075         if(this->m_vektorPredmetu.at(i) != nullptr) delete this->
      m_vektorPredmetu.at(i);
00076 }
00077
00081 void IceAge::Inventar::vypisObsah() {
00082     for(unsigned int i=0; i<this->m_vektorPredmetu.size();i++)
00083     {
00084         std::cout << i << ": " << this->m_vektorPredmetu.at(i)->getNazev() << std::endl;
00085     }
00086 }
00087
00091 unsigned int IceAge::Inventar::volba() {
00092     unsigned int x;
00093     while(true)
00094     {
00095         std::cin >> x;
00096         if(x<m_vektorPredmetu.size()) break;
00097     }
00098     return x;
00099 }
```

7.19 Dokumentace souboru cpp/Jezero.cpp

```
#include "../hlavicky/Jezero.h"
```

7.20 Jezero.cpp

```
00001
00004 #include "../hlavicky/Jezero.h"
00005
00009 IceAge::Jezero::Jezero(std::string novyNazev):Command(novyNazev) {
00010
00011 }
00012
00016 IceAge::Jezero::~Jezero() {
00017 }
00018
00022 void IceAge::Jezero::prozkoumat(IceAge::Veverka* scrat,
      std::vector<IceAge::Beast*> vektorBeast, std::vector<IceAge::Humanoid*> vektorHumanoid,
      IceAge::Reka* reka) {
00023     char volba='a';
00024     unsigned int index=0;
00025     std::cout << std::endl;
00026     std::cout << "Jsi na pokraji mensiho jezera. V prumeru je mensi jak reka, mistami vidis mokriny, ktere
       jdou prebrodit." << std::endl << std::endl;
00027     std::cout << "Na vrcholku veze vidis hnizdko." << std::endl;
00028     std::cin.ignore();
00030 std::cout << "                            .-."                      << std::endl;
00031 std::cout << "                           |/'\\\\\..-"                << std::endl;
00032 std::cout << "     _...-,,--.           '\\ /.--.\\ -.-."           << std::endl;
00033 std::cout << "  ,/'  ..:::.. \\      .-.-'/    \\\' .\\/"          << std::endl;
00034 std::cout << " /       ...:::.'\\ ,/:..| |(o)  / /o)|"           << std::endl;
00035 std::cout << "|:..   |  ..::::::'|:..  ;\\ '---'. '--'"           << std::endl;
00036 std::cout << ";::... |   .::::,/:..    .'--.   .:.'\\_"           << std::endl;
00037 std::cout << " |::.. ;  ..::'/:..   .--'    ;\\   :::.'\\"       << std::endl;
00038 std::cout << " ;::../   ..::|::.  /'          ;.  ':'.---."      << std::endl;
00039 std::cout << "  '--|    ..::;\\:.  '\\,,,____,,,/';\\. (_)  |)" << std::endl;
00040 std::cout << "     ;     ..::/:\\:.'\\|         ,__,/';----'"    << std::endl;
00041 std::cout << "    '\\       ;:.. \\: '-..      '-._,/,_,/"      << std::endl;
00042 std::cout << "      \\      ;:.   ). '\\ '>     _:\\"            << std::endl;
00043 std::cout << "       '\\,  ;:..    \\ \\ _>     </,/'\\"         << std::endl;
00044 std::cout << "          '\\|:.      '-'      <_/:.  |"          << std::endl;
00045 std::cout << "            ;:     |:.  '----':.    /"           << std::endl;
00046 std::cout << "             \\,__,/'-::....;      ,/"            << std::endl;
00047 std::cout << "              |:|           '----'"               << std::endl;
00048 std::cout << "              / < \\          /:/"                 << std::endl;
00049 std::cout << "             '-'-'-'       <:.'-._"              << std::endl;
00050 std::cout << "                           '\\,>>'"               << std::endl;
00051
00052     std::cout << std::endl;
00053     std::cout << std::endl;
00054     std::cout << std::endl;
00055
00056     std::cout << "[0] Cmuchej" <<std::endl;
00057     std::cout << "Vyber akci:";
00058     std::cin >> volba; std::cin.ignore();
00059
00060
00061     bool cyklus2=true;
00062     while(cyklus2)
00063     {
00064         switch(volba)
00065         {
00066             case '0':
00067                 cyklus2=false;
00068                 break;
00069             default:
00070                 std::cout << "Vyber akci:";
00071                 std::cin >> volba;std::cin.ignore();
00072                 break;
00073         }
00074     }
00075     cyklus2=true; volba='a';
00077     std::cout << "Orisek je urcite nekde pobliz. Citis ho." << std::endl << std::endl;
00078
00079     std::cout << "[0] Hledej dal" <<std::endl;
00080     std::cout << "Vyber akci:";
00081     std::cin >> volba;std::cin.ignore();
00082
00083     while(cyklus2)
00084     {
00085         switch(volba)
00086         {
00087             case '0':
00088                 cyklus2=false;
00089                 break;
00090             default:
00091                 std::cout << "Vyber akci:";
00092                 std::cin >> volba;std::cin.ignore();
00093                 break;
00094         }
```

```
00095     }
00096     cyklus2=true; volba='a';
00098     std::cout << std::endl << "Citis zde " << vektorBeast.at(0)->getNazev() << std::endl;
00099     std::cout << "Pach je slaby, daleko silneji ho prekryva pach orisku" << std::endl;
00100     std::cout << std::endl;
00101     std::cin.ignore();
00102     std::cout << "Rychle skaces po melcine ke skalce."<< std::endl;
00103
00105     if(scrat->getPravaRuka() != nullptr){
00106         std::cout << "Kdyz zjistis, ze se nahoru neda jen tak dostat, vztekem roztriskas svoji lodicku" <<
      std::endl;
00107         scrat->zahodPravaRuka();
00108     }
00109     else std::cout << "Nahoru se dostat neda, aspon ne s tim co mas po ruce." << std::endl;
00110
00111     std::cin.ignore();
00112     std::cout << std::endl;
00113     std::cout << "Rozhlizis se po okoli, hledaje neco, co by ti pomohlo dostat se do hnizda." << std::endl;
00114     std::cout << "V okoli se nachazi nekolik velkych vrb. Svou vyskou mohou konkurovat stromum v pralese."
      << std::endl;
00115
00116     bool cyklus=true;
00117     while(cyklus)
00118     {
00119         std::cout << std::endl;
00120         std::cout << "[" << index << "] Utrhnout si proutek" << std::endl; index++;
00121         std::cout << "[" << index << "] Utrhnout si hodne proutku a uplest z nich jedno silne lano." <<
      std::endl; index=0;
00122         std::cout << "Vyber akci:";
00123
00124         while(cyklus2)
00125         {
00126             std::cin >> volba;std::cin.ignore();
00127             std::cout << std::endl;
00128
00129             switch(volba)
00130             {
00131                 case '0':
00132                     cyklus2=false;
00133                     std::cout << "Utrhnes si jeden proutek. Kdyz zkousis jeho pevnost, tak ti praskne." <<
      std::endl << std::endl;
00134                     break;
00135                 case '1':
00136                     std::cout << "Po hodine mas dostatecne silne a dlouhe lano." << std::endl;
00137                     scrat->setPravaRuka(new IceAge::Orisek("Proutkove lano"));
00138                     cyklus2=false; cyklus=false;
00139                     break;
00140                 default:
00141                     break;
00142             }
00143         }
00144         cyklus2=true; volba='a';
00145     }
00146     cyklus2=true;
00148     std::cout << "Hazes lano na vrsek skalky a lezes nahoru." << std::endl;
00149     std::cin.ignore();
00150
00151     IceAge::Orisek *scratuvOrisek=new IceAge::Orisek("Orisek");
00153     std::cout << std::endl;
00154     std::cout << "Vylezl si nahoru. Vidis tu jen " << vektorBeast.at(1)->getNazev() << " a " <<
      scratuvOrisek->getNazev() << std::endl;
00155     std::cin.ignore();
00156     std::cout << scratuvOrisek->getNazev() << "!!!" << std::endl;
00157     std::cin.ignore();
00158     scrat->zahodPravaRuka();
00159     std::cout << "Poustis lano a pripravujes se k boji!" << std::endl;
00160     std::cin.ignore();
00161
00162     IceAge::boj(scrat,vektorBeast.at(1));
00163
00164     scrat->setPravaRuka(scratuvOrisek);
00165
00167     std::cout << "Prastil si " << vektorBeast.at(1)->getNazev() << " po hlave a utikas s ukoristenym
       oriskem." << std::endl;
00168
00169     std::cout << std::endl;
00170     std::cout << "[" << index << "] Skocit pres okraj" << std::endl; index++;
00171     std::cout << "[" << index << "] Pouzit lano k ceste dolu" << std::endl; index=0;
00172
00173
00174     while(cyklus2)
00175     {
00176         std::cout << "Vyber akci:";
00177         std::cin >> volba;std::cin.ignore();
00178             switch(volba)
00179             {
00180                 case '0':
```

```
00181                     cyklus2=false;
00182                     std::cout << "Padas do vody. V ruce uz mas jen Orisek." << std::endl << std::endl;
00183                     break;
00184                 case '1':
00185                     std::cout << "Aby si se mohl drzet lana a neztratit Orisek, hazes klacek po " <<
      vektorBeast.at(1)->getNazev() << std::endl;
00186                     std::cout << "Kdyz jsi v puli cesty, tak ti " << vektorBeast.at(1)->getNazev() << "
       precvakne lano." << std::endl;
00187                     cyklus2=false;
00188                     break;
00189                 default:
00190                     break;
00191             }
00192     }
00193     cyklus2=true;
00194     scrat->zahodLevaRuka();
00195     std::cin.ignore();
00196     std::cout << std::endl;
00198     std::cout << "Stoji proti tobe strasne moc ryb. Ocividne chteji tvuj orisek!" << std::endl;
00199
00200     std::cout << std::endl;
00201     std::cout << "[" << index << "] Utect" << std::endl; index++;
00202     std::cout << "[" << index << "] Nikdy nedostanete muj Orisek!" << std::endl; index=0;
00203
00204
00205     while(cyklus2)
00206     {
00207         std::cout << "Vyber akci:";
00208         std::cin >> volba;std::cin.ignore();
00209             switch(volba)
00210             {
00211                 case '0':
00212                     cyklus2=false;
00213                     std::cout << "Podarilo se ti uprchnout." << std::endl << std::endl;
00214                     break;
00215                 case '1':
00217                     scrat->enrageZapnout();
00218                     for(unsigned int i=2; i<vektorBeast.size(); i++){
00219                         IceAge::boj(scrat,vektorBeast.at(i));
00220                         std::cout << i-1 << ". " << vektorBeast.at(i)->getNazev() << " je omracen." <<
      std::endl;
00221                         std::this_thread::sleep_for(std::chrono::milliseconds(500));
00222                     }
00223                     scrat->enrageVypnout();
00224                     cyklus2=false;
00225                     break;
00226                 default:
00227                     break;
00228             }
00229     }
00230     std::cin.ignore();
00231
00232
00233     std::cin.ignore();
00235     std::cout << "Konecne jsi sam. Na ledovci s oriskem. Nasel sis svoje mistecko." << std::endl;
00236
00237     std::cin.ignore();
00238
00239 std::cout << "      .-----.            .'`-." << std::endl;
00240 std::cout << "    /       ,--          | .- `-." << std::endl;
00241 std::cout << "  ,'     ,-'    `.      _.-'   ,-.`.)" << std::endl;
00242 std::cout << " ;      /    ,=---`--+'      .- -. `." << std::endl;
00243 std::cout << "(   \\     ,'    =,- ,'      ( o ) | /\\" << std::endl;
00244 std::cout << " :    :  /  =,-'   /         \\-'  ;(o :" << std::endl;
00245 std::cout << "  \\  |     '     ;  (         `--'  \\ ;" << std::endl;
00246 std::cout << "   \\ |         = |  \\`--+    --.     '(" << std::endl;
00247 std::cout << "    `+        =/ :    :    `.     `.    \\" << std::endl;
00248 std::cout << "     '       =/   \\     `--. '-.    `.    `." << std::endl;
00249 std::cout << "       \\     =;      `._     : ( `-.   `.    `." << std::endl;
00250 std::cout << "        \\  = ;          `._.'   `-.-`-._\\     `-." << std::endl;
00251 std::cout << "         \\= '                      _.-'_)   (::::)" << std::endl;
00252 std::cout << "          `+       -.             `--7'   `--`..'" << std::endl;
00253 std::cout << "           (         :     .'        ;" << std::endl;
00254 std::cout << "            \\        |     |        /" << std::endl;
00255 std::cout << "             \\       | _.-|  +---'" << std::endl;
00256 std::cout << "              `--+    `.   \\\   \\\" << std::endl;
00257 std::cout << "                 /`.   '-.-\\\    `--." << std::endl;
00258 std::cout << "                /    /#### `----.'" << std::endl;
00259 std::cout << "               (  ,-'############\\\" << std::endl;
00260 std::cout << "               \\\\\/################;" << std::endl;
00261 std::cout << "                \\\################/" << std::endl;
00262 std::cout << "                 |---------------|     _.---------" << std::endl;
00263 std::cout << "                 ::::::::::::::::|_.-''" << std::endl;
00264 std::cout << "                  ::::::::::_.-''" << std::endl;
00265 std::cout << "        .-''..'---'-------''" << std::endl;
00266
00267     std::cin.ignore();
```

```
00268
00269     IceAge::MemoryArbiter::cycleDelete();
00270     exit(0);
00271 }
```

7.21 Dokumentace souboru cpp/JizniLes.cpp

```
#include "../hlavicky/JizniLes.h"
```

7.22 JizniLes.cpp

```
00001
00004 #include "../hlavicky/JizniLes.h"
00005
00009 IceAge::JizniLes::JizniLes(std::string novyNazev):
      Command(novyNazev) {
00010 }
00011
00015 IceAge::JizniLes::~JizniLes() {
00016
00017 }
00018
00022 void IceAge::JizniLes::prozkoumat(IceAge::Veverka* scrat,
      std::vector<IceAge::Beast*> vektorBeast, std::vector<IceAge::Humanoid*> vektorHumanoid,
      IceAge::Reka* reka) {
00023
00024     std::cout << "Mas silne nutkani zde zustat, ale vune ti pripomina tvuj orisek a jeho ztratu." <<
      std::endl;
00025     std::cout << "Reka tu proteka jako na planine a je stale prilis siroka na preplavani. Mozna se tu najde
       nejaka lodka." << std::endl;
00029 std::cout << "                          .-."                      << std::endl;
00030 std::cout << "                        |/'\\\\\..._"                 << std::endl;
00031 std::cout << "     _...._,,--.         '\\ /.--.\\ _.-."           << std::endl;
00032 std::cout << "  ,/'  ..:::.. \\      ..._.-'/    \\\` .\\/"         << std::endl;
00033 std::cout << " /       ...:::.'\\ ,/:..| |(o)  / /o)|"           << std::endl;
00034 std::cout << "|:..   |  ..::::::'|:..  ;\\ `---'. `--'"           << std::endl;
00035 std::cout << ";::... |   .::::,/:..    .'--.   .:.'\\_"           << std::endl;
00036 std::cout << " |::.. ;  ..::'/:..   .--'    ;\\   :::.'\\"        << std::endl;
00037 std::cout << " ;::../   ..::|::.  /'          ;.  ':'.---."      << std::endl;
00038 std::cout << "  `--|    ..::;\\:.  '\\,,,____,,,/';\\. (_)  |)" << std::endl;
00039 std::cout << "     ;     ..::/:\\:.'\\|         ,__,/';----'"    << std::endl;
00040 std::cout << "     '\\       ;:.. \\: '-..      '-._,/,_,/"      << std::endl;
00041 std::cout << "       \\      ;:.   ). '\\ '>     _:\\"           << std::endl;
00042 std::cout << "        '\\,  ;:..    \\ \\ _>     </,/'\\"        << std::endl;
00043 std::cout << "           '\\|:.      '-'      <_/:.  |"         << std::endl;
00044 std::cout << "             ;:     |:.  '----':.    /"          << std::endl;
00045 std::cout << "              \\,__,/'-::....;      ,/"           << std::endl;
00046 std::cout << "               |:|           '----'"              << std::endl;
00047 std::cout << "               / < \\          /:/"                << std::endl;
00048 std::cout << "              '-'-'-'       <:.'-._"              << std::endl;
00049 std::cout << "                            '\\,>>'"              << std::endl;
00050
00051 std::cout << std::endl;
00052 std::cout << std::endl;
00053 std::cout << std::endl;
00054
00055
00056     bool cyklus2=true;
00057     unsigned int index=0;
00058     unsigned int volbaCislo=0;
00059     char volba='a';
00061     while(cyklus2)
00062     {
00063             std::cout << std::endl;
00064             std::cout << "[" << index << "] Odejit" <<std::endl; index++;
00065             std::cout << "[" << index << "] Napit se " <<std::endl; index++;
00066             std::cout << "[" << index << "] Pokusit se preplavat na druhou stranu svepomoci" <<std::endl;
      index++;
00067             std::cout << "[" << index << "] Zautocit na nekoho" <<std::endl; index++;
00068             std::cout << "[" << index << "] Sebrat klacek" <<std::endl; index++;
00069             std::cout << "[" << index << "] Vyrobit lodku" <<std::endl;
00070
00071             index=0;
00072             std::cout << "Vyber akci:";
```

```
00073             std::cin >> volba;std::cin.ignore();
00074             switch(volba)
00075             {
00076                 case '0':
00077                     return;
00078                     break;
00079                 case '1':
00080                     scrat->changeZivotAktualni(reka->
      getHodnotaZivota());
00081                     std::cout << "Aktualne mas " << scrat->getZivotAktualni() << "/" <<
      scrat->getZivotMaximalni() << std::endl;
00082                     continue;
00083                     break;
00084                 case '2':
00085                     std::cout << "Proud je moc silny, utopil ses. Nedokazal si ziskat zpatky orisek!" <<
      std::endl;
00086                     IceAge::MemoryArbiter::cycleDelete();
00087                     exit(0);
00088                     break;
00089                 case '3':
00090                     for(unsigned int i=0; i<vektorBeast.size(); index++,i++)
00091                         std::cout << "[" << index << "] " << vektorBeast.at(i)->getNazev() << std::endl;
00092
00093
00094                     for(unsigned int j=0; j<vektorHumanoid.size(); index++, j++)
00095                         std::cout << "[" << index << "] " << vektorHumanoid.at(j)->getNazev() << std::endl;
00096
00097                     std::cout << "Vyber si cil: ";
00098
00099                     while(true){
00101                         std::cin >> volbaCislo;std::cin.ignore();
00102
00103                         if(volbaCislo < index){
00104                             if(volbaCislo<vektorBeast.size()) boj(scrat,vektorBeast.at(volbaCislo));
00105                             else boj(scrat, vektorHumanoid.at(volbaCislo-vektorBeast.size()));
00106                             break;
00107                         }
00108                     }
00109                     index=0;
00110                     break;
00111                 case '4':
00112                         if(scrat->getLevaRuka() == nullptr) scrat->
      setLevaRuka(new IceAge::Zbran("Maly klacik",3));
00113                         else std::cout << "Uz mas Maly klacik. Vic neuneses." << std::endl;
00114                     break;
00115                 case '5':
00116                         if(scrat->getPravaRuka() == nullptr) scrat->
      setPravaRuka(new IceAge::Orisek("Lodka"));
00117                         else std::cout << "Uz mas Lodku. Vic neuneses." << std::endl;
00118                     break;
00119                 default:
00120                     break;
00121             }
00122         }
00123 }
```

7.23 Dokumentace souboru cpp/Ledovec.cpp

```
#include "../hlavicky/Ledovec.h"
```

7.24 Ledovec.cpp

```
00001
00004 #include "../hlavicky/Ledovec.h"
00005
00009 IceAge::Ledovec::Ledovec(std::string novyNazev):Command(novyNazev) {
00010 }
00011
00015 IceAge::Ledovec::~Ledovec() {
00016 }
00017
00021 void IceAge::Ledovec::prozkoumat(IceAge::Veverka* scrat,
      std::vector<IceAge::Beast*> vektorBeast, std::vector<IceAge::Humanoid*> vektorHumanoid,
      IceAge::Reka* reka) {
```

```
00022     std::cout << std::endl;
00023     std::cout << "Ledovec se zhroutil, krome laviny do udoli tu neni nikdo a nic." << std::endl;
00024     std::cout << "Spadl na tebe kus ledu! Ztracis 3 zivoty!" <<std::endl;
00025 //    std::cout << vektorBeast.size() << std::endl;
00026     scrat->changeZivotAktualni(-3);
00027     std::cout << "Vidis " << vektorBeast.at(0)->getNazev() << ". Udelal si z tebe cvicny terc." <<
      std::endl;
00028     std::cout << std::endl;
00029 }
```

7.25 Dokumentace souboru cpp/LocationDirector.cpp

```
#include "../hlavicky/LocationDirector.h"
```

7.26 LocationDirector.cpp

```
00001
00004 #include "../hlavicky/LocationDirector.h"
00005
00009 IceAge::LocationDirector::LocationDirector(
      IceAge::BuilderLokace* novyBuilder) {
00010     this->setBuilder(novyBuilder);
00011 }
00012
00016 IceAge::Lokace* IceAge::LocationDirector::createLokace(
      ) {
00017     this->m_builder->createLokace();
00018     return this->m_builder->getLokace();
00019 }
00020
00024 void IceAge::LocationDirector::setBuilder(
      IceAge::BuilderLokace* novyBuilder) {
00025     this->m_builder=novyBuilder;
00026 }
00027
00031 IceAge::LocationDirector::~LocationDirector(){
00032     if(this->m_builder != nullptr) delete this->m_builder;
00033 }
```

7.27 Dokumentace souboru cpp/Lokace.cpp

```
#include "../hlavicky/Lokace.h"
```

7.28 Lokace.cpp

```
00001
00004 #include "../hlavicky/Lokace.h"
00005
00006 IceAge::Lokace::Lokace(std::string novyNazev, std::string novyPopis):
      IceAge::Nazev(novyNazev), Popis(novyPopis) {
00007     this->m_odkazReka=nullptr;
00008     this->m_prikaz=nullptr;
00009 }
00010
00011 void IceAge::Lokace::pridejBeast(IceAge::Beast* novyBeast) {
00012     this->m_vektorBeast.push_back(novyBeast);
00013 }
00014
00015 void IceAge::Lokace::pridejHumanoid(
      IceAge::Humanoid* novyHumanoid) {
00016     this->m_vektorHumanoid.push_back(novyHumanoid);
```

```
00017 }
00018
00019 void IceAge::Lokace::smazBeast(unsigned int index) {
00020     IceAge::Beast* third=this->m_vektorBeast.at(index);
00021     this->m_vektorBeast.at(index)=this->m_vektorBeast.at(this->
      m_vektorBeast.size()-1);
00022     this->m_vektorBeast.at(this->m_vektorBeast.size()-1)=third;
00023     this->m_vektorBeast.pop_back();
00024 }
00025
00026 void IceAge::Lokace::smazHumanoid(unsigned int index) {
00027     IceAge::Humanoid* third=this->m_vektorHumanoid.at(index);
00028     this->m_vektorHumanoid.at(index)=this->m_vektorHumanoid.at(this->
      m_vektorHumanoid.size()-1);
00029     this->m_vektorHumanoid.at(this->m_vektorHumanoid.size()-1)=third;
00030     this->m_vektorHumanoid.pop_back();
00031 }
00032
00033 void IceAge::Lokace::pridejReku() {
00034     this->m_odkazReka=new IceAge::Reka(LECIVOST_REKY);
00035 }
00036
00037 void IceAge::Lokace::smazReku() {
00038     delete this->m_odkazReka;
00039 }
00040
00041 void IceAge::Lokace::pridejLokaci(IceAge::Lokace* novaLokace) {
00042     this->m_vektorLokaci.push_back(novaLokace);
00043 }
00044
00045 void IceAge::Lokace::smazLokaci(unsigned int index) {
00046     IceAge::Lokace* third=this->m_vektorLokaci.at(index);
00047     this->m_vektorLokaci.at(index)=this->m_vektorLokaci.at(this->
      m_vektorLokaci.size()-1);
00048     this->m_vektorLokaci.at(this->m_vektorLokaci.size()-1)=nullptr;
00049     this->m_vektorLokaci.pop_back();
00050     delete third;
00051 }
00052
00053 void IceAge::Lokace::odeberLokaci(unsigned int index) {
00054     IceAge::Lokace* third=this->m_vektorLokaci.at(index);
00055     this->m_vektorLokaci.at(index)=this->m_vektorLokaci.at(this->
      m_vektorLokaci.size()-1);
00056     this->m_vektorLokaci.at(this->m_vektorLokaci.size()-1)=third;
00057     this->m_vektorLokaci.pop_back();
00058 }
00059
00060 void IceAge::Lokace::vypisLokace() {
00061     for(unsigned int i=0; i<this->m_vektorLokaci.size(); i++)
00062     {
00063         std::cout << "[" << i << "] " << this->m_vektorLokaci.at(i)->getNazev() << std::endl;
00064     }
00065 }
00066
00067 unsigned int IceAge::Lokace::getVelikostVektoruLokaci(){
00068     return this->m_vektorLokaci.size();
00069 }
00070
00071 std::string IceAge::Lokace::getNazevLokace(){
00072     return this->getNazev();
00073 }
00074
00075 std::string IceAge::Lokace::getPopisLokace(){
00076     return this->getPopis();
00077 }
00078
00079 IceAge::Beast* IceAge::Lokace::getBeast(unsigned int index){
00080     if(index<this->m_vektorBeast.size()) return this->m_vektorBeast.at(index);
00081     else return this->m_vektorBeast.back();
00082 }
00083
00084 IceAge::Humanoid* IceAge::Lokace::getHumanoid(unsigned int index
      ){
00085     if(index<this->m_vektorHumanoid.size()) return this->
      m_vektorHumanoid.at(index);
00086     else return this->m_vektorHumanoid.back();
00087 }
00088
00089 void IceAge::Lokace::setPrikaz(){
00090     if(this->getNazevLokace() == "Ledovec")                  this->
      m_prikaz=new IceAge::Ledovec("Ledovec");
00091     else if(this->getNazevLokace() == "Zapadni planina")     this->
      m_prikaz=new IceAge::ZapadniPlanina("Zapadni planina");
00092     else if(this->getNazevLokace() == "Zapadni breh reky")   this->
      m_prikaz=new IceAge::ZapadniBrehReky("Zapadni breh reky");
00093     else if(this->getNazevLokace() == "Jizni les")           this->
      m_prikaz=new IceAge::JizniLes("Jizni les");
```

```
00094     else if(this->getNazevLokace() == "Severni les")         this->
      m_prikaz=new IceAge::SeverniLes("Severni les");
00095     else if(this->getNazevLokace() == "Vychodni planina")    this->
      m_prikaz=new IceAge::VychodniPlanina("Vychodni planina");
00096     else if(this->getNazevLokace() == "Vychodni breh reky")  this->
      m_prikaz=new IceAge::VychodniBrehReky("Vychodni breh reky");
00097     else if(this->getNazevLokace() == "Jezero")              this->
      m_prikaz=new IceAge::Jezero("Jezero");
00098
00099 }
00100
00101 IceAge::Lokace::~Lokace(){
00102     if(this->m_odkazReka != nullptr) delete this->m_odkazReka;
00103     if(this->m_prikaz != nullptr) delete this->m_prikaz;
00104
00105     for(unsigned int i=0; i<m_vektorBeast.size(); i++)
00106         if(this->m_vektorBeast.at(i) != nullptr) delete this->
      m_vektorBeast.at(i);
00107
00108     for(unsigned int j=0; j<m_vektorHumanoid.size(); j++)
00109         if(this->m_vektorHumanoid.at(j) != nullptr) delete this->
      m_vektorHumanoid.at(j);
00110
00111  /*   for(unsigned int i=0; i<m_vektorLokaci.size(); i++)
00112         if(this->m_vektorLokaci.at(i) != nullptr) delete this->m_vektorLokaci.at(i);*/
00113 }
00114
00115 IceAge::Lokace* IceAge::Lokace::getLokace(unsigned int index){
00116     if(index < this->m_vektorLokaci.size()) return this->
      m_vektorLokaci.at(index);
00117     else return this;
00118 }
00119
00120 void IceAge::Lokace::prozkoumatLokaci(
      IceAge::Veverka* scrat){
00121     this->m_prikaz->prozkoumat(scrat, this->getVektorBeast(), this->
      getVektorHumanoid(), this->getReka());
00122 }
00123
00124 std::vector<IceAge::Beast*> IceAge::Lokace::getVektorBeast(){
00125     return this->m_vektorBeast;
00126 }
00127
00128 std::vector<IceAge::Humanoid*> IceAge::Lokace::getVektorHumanoid(){
00129     return this->m_vektorHumanoid;
00130 }
00131
00132 IceAge::Reka* IceAge::Lokace::getReka(){
00133     return this->m_odkazReka;
00134 }
00135
00136 /*unsigned int IceAge::Lokace::volba() {
00137
00138 }*/
```

7.29 Dokumentace souboru cpp/Memory.cpp

```
#include "../hlavicky/Memory.h"
```

7.30 Memory.cpp

```
00001
00004 #include "../hlavicky/Memory.h"
00005
00009 unsigned int IceAge::Memory::getId() {
00010     return this->m_id;
00011 }
00012
00016 void IceAge::Memory::setId(unsigned int noveId) {
00017     this->m_id=noveId;
00018 }
00019
00023 IceAge::Memory::Memory() {
00024     IceAge::MemoryArbiter::add(this);
```

```
00025 }
00026
00030 IceAge::Memory::~Memory() {
00031     IceAge::MemoryArbiter::remove(this->getId());
00032 }
```

7.31 Dokumentace souboru cpp/MemoryArbiter.cpp

```
#include "../hlavicky/MemoryArbiter.h"
```

7.32 MemoryArbiter.cpp

```
00001
00004 #include "../hlavicky/MemoryArbiter.h"
00005
00006
00007 std::vector<IceAge::Memory*> IceAge::MemoryArbiter::s_memoryVector;
00012 IceAge::MemoryArbiter::MemoryArbiter() {
00013 }
00014
00018 /*static*/ void IceAge::MemoryArbiter::add(
      IceAge::Memory* target) {
00019     s_memoryVector.push_back(target);
00020     s_memoryVector.at(s_memoryVector.size()-1)->setId(
      s_memoryVector.size()-1);
00021 }
00022
00026 /*static*/ void IceAge::MemoryArbiter::remove(unsigned int targetId) {
00027
00028     s_memoryVector.at(targetId)=nullptr;
00029     s_memoryVector.erase(s_memoryVector.begin()+targetId,
      s_memoryVector.begin()+targetId);
00030 }
00031
00035 /*static*/ void IceAge::MemoryArbiter::cycleDelete() {
00036     for(unsigned int i=0; i<s_memoryVector.size();i++){
00037         if(s_memoryVector.at(i) != nullptr) delete s_memoryVector.at(i);
00038     }
00039 }
```

7.33 Dokumentace souboru cpp/Nazev.cpp

```
#include "../hlavicky/Nazev.h"
```

7.34 Nazev.cpp

```
00001
00004 #include "../hlavicky/Nazev.h"
00005
00006
00010 std::string IceAge::Nazev::getNazev() {
00011     return this->m_nazev;
00012 }
00013
00017 void IceAge::Nazev::setNazev(std::string novyNazev) {
00018     this->m_nazev=novyNazev;
00019 }
00020
00024 IceAge::Nazev::Nazev(std::string novyNazev) {
00025     this->setNazev(novyNazev);
00026 }
```

7.35 Dokumentace souboru cpp/ObleceneVybaveniHumanoid.cpp

```
#include "../hlavicky/ObleceneVybaveniHumanoid.h"
```

7.36 ObleceneVybaveniHumanoid.cpp

```
00001
00004 #include "../hlavicky/ObleceneVybaveniHumanoid.h"
00005
00009 IceAge::Predmet* IceAge::ObleceneVybaveniHumanoid::getLevaRuka
      () {
00010     return this->m_levaRuka;
00011 }
00012
00016 void IceAge::ObleceneVybaveniHumanoid::setLevaRuka(
      IceAge::Predmet* novaLevaRuka) {
00017     this->m_levaRuka=novaLevaRuka;
00018 }
00019
00023 IceAge::Predmet* IceAge::ObleceneVybaveniHumanoid::getPravaRuka
      () {
00024     return this->m_pravaRuka;
00025 }
00026
00030 void IceAge::ObleceneVybaveniHumanoid::setPravaRuka(
      IceAge::Predmet* novaPravaRuka) {
00031     this->m_pravaRuka=novaPravaRuka;
00032 }
00033
00037 IceAge::Predmet* IceAge::ObleceneVybaveniHumanoid::getZada
      () {
00038     return this->m_zada;
00039 }
00040
00044 void IceAge::ObleceneVybaveniHumanoid::setZada(
      IceAge::Predmet* novaZada) {
00045     this->m_zada=novaZada;
00046 }
00047
00051 IceAge::ObleceneVybaveniHumanoid::ObleceneVybaveniHumanoid
      (IceAge::Predmet* levaRuka, IceAge::Predmet* pravaRuka,
      IceAge::Predmet* zada) {
00052     this->setLevaRuka(levaRuka);
00053     this->setPravaRuka(pravaRuka);
00054     this->setZada(zada);
00055 }
00056
00060 IceAge::ObleceneVybaveniHumanoid::~ObleceneVybaveniHumanoid
      (){
00061     if(this->m_levaRuka != nullptr) delete this->m_levaRuka;
00062     if(this->m_pravaRuka != nullptr) delete this->m_pravaRuka;
00063     if(this->m_zada != nullptr) delete this->m_zada;
00064 }
```

7.37 Dokumentace souboru cpp/ObleceneVybaveniVeverky.cpp

```
#include "../hlavicky/ObleceneVybaveniVeverky.h"
```

7.38 ObleceneVybaveniVeverky.cpp

```
00001
00004 #include "../hlavicky/ObleceneVybaveniVeverky.h"
00005
00009 IceAge::Predmet* IceAge::ObleceneVybaveniVeverky::getLevaRuka
      () {
```

```
00010     return this->m_levaRuka;
00011 }
00012
00016 void IceAge::ObleceneVybaveniVeverky::setLevaRuka(
      IceAge::Predmet* novaLevaRuka) {
00017     this->m_levaRuka=novaLevaRuka;
00018 }
00019
00023 IceAge::Predmet* IceAge::ObleceneVybaveniVeverky::getPravaRuka
      () {
00024     return this->m_pravaRuka;
00025 }
00026
00030 void IceAge::ObleceneVybaveniVeverky::setPravaRuka(
      IceAge::Predmet* novaPravaRuka) {
00031     this->m_pravaRuka=novaPravaRuka;
00032 }
00033
00037 IceAge::ObleceneVybaveniVeverky::ObleceneVybaveniVeverky
      (IceAge::Predmet* novaLevaRuka, IceAge::Predmet* novaPravaRuka) {
00038     this->setLevaRuka(novaLevaRuka);
00039     this->setPravaRuka(novaPravaRuka);
00040 }
00041
00045 IceAge::ObleceneVybaveniVeverky::~ObleceneVybaveniVeverky
      (){
00046     if(this->m_levaRuka != nullptr) delete this->m_levaRuka;
00047     if(this->m_pravaRuka != nullptr) delete this->m_pravaRuka;
00048 }
00049
00053 void IceAge::ObleceneVybaveniVeverky::zahodLevaRuka(){
00054     if(this->m_levaRuka != nullptr) delete this->m_levaRuka;
00055     this->m_levaRuka=nullptr;
00056 }
00057
00061 void IceAge::ObleceneVybaveniVeverky::zahodPravaRuka(){
00062     if(this->m_pravaRuka != nullptr) delete this->m_pravaRuka;
00063     this->m_pravaRuka=nullptr;
00064 }
```

7.39 Dokumentace souboru cpp/Orisek.cpp

```
#include "../hlavicky/Orisek.h"
```

7.40 Orisek.cpp

```
00001
00004 #include "../hlavicky/Orisek.h"
00005
00009 IceAge::Orisek::Orisek(std::string novyNazev):Predmet(novyNazev) {
00010     this->setTyp('n');
00011 }
00012
00016 IceAge::Orisek::~Orisek(){
00017 }
00018
00022  short IceAge::Orisek::getAtribut(){
00023     return 0;
00024  }
```

7.41 Dokumentace souboru cpp/Popis.cpp

```
#include "../hlavicky/Popis.h"
```

7.42 Popis.cpp

```
00001
00004 #include "../hlavicky/Popis.h"
00005
00009 std::string IceAge::Popis::getPopis() {
00010     return this->m_popis;
00011 }
00012
00016 void IceAge::Popis::setPopis(std::string novyPopis) {
00017     this->m_popis=novyPopis;
00018 }
00019
00023 IceAge::Popis::Popis(std::string novyPopis) {
00024     this->setPopis(novyPopis);
00025 }
00026
00030 IceAge::Popis::~Popis(){
00031 }
```

7.43 Dokumentace souboru cpp/Predmet.cpp

```
#include "../hlavicky/Predmet.h"
```

7.44 Predmet.cpp

```
00001
00004 #include "../hlavicky/Predmet.h"
00008 char IceAge::Predmet::getTyp() {
00009     return this->m_typ;
00010 }
00011
00015 void IceAge::Predmet::setTyp(char novyTyp) {
00016     if(/*novyTyp == 'c' || */novyTyp == 'w' || novyTyp == 'n') this->m_typ=novyTyp;
00017     else this->m_typ='u'; //unknown
00018 }
00019
00023 IceAge::Predmet::Predmet(std::string novyNazev):Nazev(novyNazev) {
00024 }
00025
00029 IceAge::Predmet::~Predmet(){
00030 }
```

7.45 Dokumentace souboru cpp/Reka.cpp

```
#include "../hlavicky/Reka.h"
```

7.46 Reka.cpp

```
00001
00004 #include "../hlavicky/Reka.h"
00005
00009 short IceAge::Reka::getHodnotaZivota() {
00010     return this->m_hodnotaZivota;
00011 }
00012
00016 void IceAge::Reka::setHodnotaZivota(short novaHodnotaZivota) {
00017     if(novaHodnotaZivota>0) this->m_hodnotaZivota=novaHodnotaZivota;
00018 }
00019
00023 IceAge::Reka::Reka(short novaHodnota) {
00024     this->setHodnotaZivota(novaHodnota);
00025 }
00026
00030 IceAge::Reka::~Reka(){
00031 }
```

7.47 Dokumentace souboru cpp/SeverniLes.cpp

```
#include "../hlavicky/SeverniLes.h"
```

7.48 SeverniLes.cpp

```
00001
00004 #include "../hlavicky/SeverniLes.h"
00005
00009 IceAge::SeverniLes::SeverniLes(std::string novyNazev):
      Command(novyNazev) {
00010 }
00011
00015 IceAge::SeverniLes::~SeverniLes() {
00016 }
00017
00021 void IceAge::SeverniLes::prozkoumat(
      IceAge::Veverka* scrat, std::vector<IceAge::Beast*> vektorBeast,
      std::vector<IceAge::Humanoid*> vektorHumanoid, IceAge::Reka* reka) {
00022     std::cout << std::endl;
00023     std::cout << "Oproti lesu na druhe strane, tento prales je plny svinciku a to i na zvireci pomery." <<
      std::endl;
00024     std::cout << "Ocividne teto oblasti vevodi tlupa velkych dvounozcu a jejich rivalove savlozubi tygri."
      <<std::endl;
00025
00026     bool cyklus2=true;
00027     unsigned int index=0;
00028     char volba='a';
00030     while(cyklus2)
00031     {
00032         std::cout << std::endl;
00033         std::cout << "[" << index << "] Nenapadne se vytratit" <<std::endl; index++;
00034         std::cout << "[" << index << "] Napit se " <<std::endl; index=0;
00035
00036         std::cout << "Vyber akci:";
00037         std::cin >> volba;std::cin.ignore();
00038
00039         switch(volba){
00040             case '0':
00041                     std::cout << "Nepovedlo se, vsiml si te " << vektorHumanoid.at(0)->getNazev() <<
      std::endl;
00042                     IceAge::boj(vektorHumanoid.at(0), scrat);
00043                     std::cout << "Povedlo se ti dostat do bezpeci, ale ztratil si svou lodku" << std::endl;
00044                     scrat->zahodPravaRuka();
00045                     cyklus2=false;
00046                 break;
00047             case '1':
00048                 scrat->changeZivotAktualni(reka->
      getHodnotaZivota());
00049                 std::cout << "Aktualne mas " << scrat->getZivotAktualni() << "/" << scrat->
      getZivotMaximalni() << std::endl;
00050                 continue;
00051                 break;
00052         }
00053     }
00054 }
```

7.49 Dokumentace souboru cpp/Staty.cpp

```
#include "../hlavicky/Staty.h"
```

7.50 Staty.cpp

```
00001
00004 #include "../hlavicky/Staty.h"
00005
00009 short IceAge::Staty::getSila() {
00010     return this->m_sila;
00011 }
00012
00016 void IceAge::Staty::setSila(short novaSila) {
00017     this->m_sila=novaSila;
00018 }
00019
00023 short IceAge::Staty::getOdolnost() {
00024     return this->m_odolnost;
00025 }
00026
00030 void IceAge::Staty::setOdolnost(short novaOdolnost) {
00031     this->m_odolnost=novaOdolnost;
00032 }
00033
00037 IceAge::Staty::Staty(short novaSila, short novaOdolnost) {
00038     this->setSila(novaSila);
00039     this->setOdolnost(novaOdolnost);
00040 }
00041
00045 IceAge::Staty::~Staty(){
00046 }
```

7.51 Dokumentace souboru cpp/Veverka.cpp

```
#include "../hlavicky/Veverka.h"
```

7.52 Veverka.cpp

```
00001
00004 #include "../hlavicky/Veverka.h"
00005
00006 IceAge::Veverka* IceAge::Veverka::s_vlastniInstance =
      nullptr;
00011 IceAge::Veverka::Veverka(short novyZivot, short novaSila, short novaOdolnost):
      ObleceneVybaveniVeverky(nullptr, nullptr),
00012                                                 Staty(novaSila,novaOdolnost),
00013                                                 Zivot(novyZivot, novyZivot) {
00014     this->setPravaRuka(new IceAge::Orisek("Orisek"));
00015 }
00019 /*static*/ IceAge::Veverka* IceAge::Veverka::getInstance() {
00020     if(s_vlastniInstance == nullptr) s_vlastniInstance = new
      IceAge::Veverka(SCRAT_HP,SCRAT_SILA,
      SCRAT_ODOLNOST);
00021     return s_vlastniInstance;
00022 }
00023
00027 short IceAge::Veverka::utok(){
00028     short utokCelkem=0;
00029     if(this->m_levaRuka != nullptr)     if(this->m_levaRuka->
      getTyp() == 'w') utokCelkem+=this->m_levaRuka->getAtribut();
00030     if(this->m_pravaRuka != nullptr)    if(this->m_pravaRuka->
      getTyp() == 'w') utokCelkem+=this->m_pravaRuka->getAtribut();
00031 //    std::cout << this->getSila() << " " << utokCelkem << std::endl;
00032     return this->getSila() + utokCelkem;
00033 }
00034
00038 short IceAge::Veverka::obrana(){
00039 //    short obranaCelkem=0;
00040 /*
00041     if(this->m_levaRuka != nullptr)     if(this->m_levaRuka->getTyp() == 'c')
       obranaCelkem+=this->m_levaRuka->getAtribut();
00042     if(this->m_pravaRuka != nullptr)    if(this->m_pravaRuka->getTyp() == 'c')
       obranaCelkem+=this->m_pravaRuka->getAtribut();
00043 */
00044     return this->getOdolnost()/* + obranaCelkem*/;
```

```
00045 }
00046
00050 IceAge::Veverka::~Veverka(){
00051 }
00052
00056 void IceAge::Veverka::enrageZapnout(){
00057     this->setSila(40);
00058     this->setOdolnost(40);
00059     std::cout << "***ENRAGE***" << std::endl;
00060 }
00061
00065 void IceAge::Veverka::enrageVypnout(){
00066     this->setSila(SCRAT_SILA);
00067     this->setOdolnost(SCRAT_ODOLNOST);
00068     std::cout << "Uz si to rozdychal." << std::endl;
00069 }
```

7.53 Dokumentace souboru cpp/VolneFunkce.cpp

```
#include "../hlavicky/VolneFunkce.h"
```

7.54 VolneFunkce.cpp

```
00001
00004 #include "../hlavicky/VolneFunkce.h"
00005
00012 void IceAge::boj(IceAge::Zivot* utocnik, IceAge::Zivot* obrance) {
00013     std::cout << std::endl;
00014     if(utocnik->utok() > obrance->obrana()){
00015         std::cout << "Obrance ztraci "<< utocnik->utok() - obrance->obrana() << " zivotu" <<
      std::endl;
00016         obrance->changeZivotAktualni(obrance->obrana() - utocnik->
      utok());
00017     }
00018     if(obrance->utok() > utocnik->obrana()){
00019         std::cout << "Utocnik ztraci "<< obrance->utok() - utocnik->obrana() << " zivotu" <<
      std::endl;
00020         utocnik->changeZivotAktualni(utocnik->obrana() - obrance->
      utok());
00021     }
00022     std::cout << "Stav zivotu:" << std::endl;
00023     std::cout << "************" << std::endl;
00024     std::cout << "Utocnik: " << utocnik->getZivotAktualni() << "/" << utocnik->
      getZivotMaximalni() << std::endl;
00025     std::cout << "Obrance: " << obrance->getZivotAktualni() << "/" << obrance->
      getZivotMaximalni() << std::endl;
00026 }
```

7.55 Dokumentace souboru cpp/VychodniBrehReky.cpp

```
#include "../hlavicky/VychodniBrehReky.h"
```

7.56 VychodniBrehReky.cpp

```
00001
00004 #include "../hlavicky/VychodniBrehReky.h"
00005
00009 IceAge::VychodniBrehReky::VychodniBrehReky(std::string novyNazev)
      :Command(novyNazev) {
00010 }
00011
```

```
00015 IceAge::VychodniBrehReky::~VychodniBrehReky() {
00016 }
00017
00021 void IceAge::VychodniBrehReky::prozkoumat(
      IceAge::Veverka* scrat, std::vector<IceAge::Beast*> vektorBeast,
      std::vector<IceAge::Humanoid*> vektorHumanoid, IceAge::Reka* reka) {
00022     std::cout << std::endl;
00023     std::cout << "Pomalu si prisel k rece. Jsi na druhe strane reky, u ktere si ztratil Supomuta." <<
      std::endl;
00024
00025     bool cyklus2=true;
00026     unsigned int index=0;
00027     unsigned int volbaCislo=0;
00028     char volba='a';
00030     while(cyklus2)
00031         {
00032             std::cout << std::endl;
00033             std::cout << "[" << index << "] Odejit" <<std::endl; index++;
00034             std::cout << "[" << index << "] Napit se " <<std::endl; index++;
00035             std::cout << "[" << index << "] Pokusit se preplavat na druhou stranu" <<std::endl; index++;
00036             std::cout << "[" << index << "] Zautocit na nekoho" <<std::endl; index++;
00037             std::cout << "[" << index << "] Hledat orisek cichem" <<std::endl; index=0;
00038             std::cout << "Vyber akci:";
00039             std::cin >> volba;std::cin.ignore();
00040             switch(volba)
00041             {
00042                 case '0':
00043                     return;
00044                     break;
00045                 case '1':
00046                     scrat->changeZivotAktualni(reka->
      getHodnotaZivota());
00047                     std::cout << "Aktualne mas " << scrat->getZivotAktualni() << "/" <<
      scrat->getZivotMaximalni() << std::endl;
00048                     continue;
00049                     break;
00050                 case '2':
00051                     std::cout << "Proud je moc silny, utopil ses. Nedokazal si ziskat zpatky orisek!" <<
      std::endl;
00052                     IceAge::MemoryArbiter::cycleDelete();
00053                     exit(0);
00054                     break;
00055                 case '3':
00056                     for(unsigned int i=0; i<vektorBeast.size(); index++,i++)
00057                         std::cout << "[" << index << "] " << vektorBeast.at(i)->getNazev() << " - "<<
      vektorBeast.at(i)->getPopis() << std::endl;
00058
00059
00060                     for(unsigned int j=0; j<vektorHumanoid.size(); index++, j++)
00061                         std::cout << "[" << index << "] " << vektorHumanoid.at(j)->getNazev() << " - "<<
      vektorHumanoid.at(j)->getPopis() << std::endl;
00062
00063                     std::cout << "Vyber si cil: ";
00064
00065                     while(true){
00067                         std::cin >> volbaCislo;std::cin.ignore();
00068
00069                         if(volbaCislo < index){
00070                             if(volbaCislo<vektorBeast.size()) boj(scrat,vektorBeast.at(volbaCislo));
00071                             else boj(scrat, vektorHumanoid.at(volbaCislo-vektorBeast.size()));
00072                             break;
00073                         }
00074                     }
00075                     index=0;
00076                     break;
00077                 case '4':
00078                     std::cout << std::endl;
00079                     std::cout << "Zapojil si svuj cich, orisek je smerem k jezeru!" << std::endl;
00080                     break;
00081                 default:
00082                     break;
00083             }
00084         }
00085 }
```

7.57 Dokumentace souboru cpp/VychodniPlanina.cpp

```
#include "../hlavicky/VychodniPlanina.h"
```

7.58 VychodniPlanina.cpp

```
00001
00004 #include "../hlavicky/VychodniPlanina.h"
00005
00009 IceAge::VychodniPlanina::VychodniPlanina(std::string novyNazev):
      Command(novyNazev) {
00010 }
00011
00015 IceAge::VychodniPlanina::~VychodniPlanina() {
00016 }
00017
00021 void IceAge::VychodniPlanina::prozkoumat(
      IceAge::Veverka* scrat, std::vector<IceAge::Beast*> vektorBeast,
      std::vector<IceAge::Humanoid*> vektorHumanoid, IceAge::Reka* reka) {
00022     std::cout << std::endl;
00023     std::cout << "Jen slysis \"Hele maso!\" a uz vis, ze je zle" << std::endl;
00024
00025     bool cyklus2=true;
00026     unsigned int index=0;
00027     char volba='a';
00029     while(cyklus2)
00030     {
00031             std::cout << std::endl;
00032             std::cout << "[" << index << "] Zdrhej" <<std::endl; index=0;
00033
00034             std::cout << "Vyber akci:";
00035             std::cin >> volba;std::cin.ignore();
00036             switch(volba)
00037             {
00038                 case '0':
00039                     if(scrat->getPravaRuka() != nullptr){
00040                         std::cout << "Zahazujes lodku a utikas" << std::endl;
00041                         scrat->zahodPravaRuka();
00042                     }
00043                     std::cout << "Uspesne si utekl " << vektorHumanoid.at(0)->getPopis() << std::endl;
00044                     std::cout << std::endl;
00045                     return;
00046                     break;
00047                 default:
00048                     break;
00049
00050             }
00051         }
00052 }
```

7.59 Dokumentace souboru cpp/ZapadniBrehReky.cpp

```
#include "../hlavicky/ZapadniBrehReky.h"
```

7.60 ZapadniBrehReky.cpp

```
00001
00004 #include "../hlavicky/ZapadniBrehReky.h"
00005
00009 IceAge::ZapadniBrehReky::ZapadniBrehReky(std::string novyNazev):
      Command(novyNazev) {
00010 }
00011
00015 IceAge::ZapadniBrehReky::~ZapadniBrehReky() {
00016 }
00017
00021 void IceAge::ZapadniBrehReky::prozkoumat(
      IceAge::Veverka* scrat, std::vector<IceAge::Beast*> vektorBeast,
      std::vector<IceAge::Humanoid*> vektorHumanoid, IceAge::Reka* reka) {
00022
00023     std::cout << "Pomalu si prisel k rece. Na druhou stranu to je nejmene 100 metru." << std::endl;
00024
00025
00026     bool cyklus2=true;
00027     unsigned int index=0;
00028     unsigned int volbaCislo=0;
```

```
00029     char volba='a';
00031     while(cyklus2)
00032         {
00033             std::cout << std::endl;
00034             std::cout << "[" << index << "] Odejit" <<std::endl; index++;
00035             std::cout << "[" << index << "] Napit se " <<std::endl; index++;
00036             std::cout << "[" << index << "] Pokusit se preplavat na druhou stranu" <<std::endl; index++;
00037             std::cout << "[" << index << "] Zautocit na nekoho" <<std::endl; index=0;
00038             std::cout << "Vyber akci:";
00039             std::cin >> volba;std::cin.ignore();
00040             switch(volba)
00041             {
00042                 case '0':
00043                     return;
00044                     break;
00045                 case '1':
00046                     scrat->changeZivotAktualni(reka->
      getHodnotaZivota());
00047                     std::cout << "Aktualne mas " << scrat->getZivotAktualni() << "/" <<
      scrat->getZivotMaximalni() << std::endl;
00048                     continue;
00049                     break;
00050                 case '2':
00051                     std::cout << "Proud je moc silny, utopil ses. Nedokazal si ziskat zpatky orisek!" <<
      std::endl;
00052                     IceAge::MemoryArbiter::cycleDelete();
00053                     exit(0);
00054                     break;
00055                 case '3':
00056                     for(unsigned int i=0; i<vektorBeast.size(); index++,i++)
00057                          std::cout << "[" << index << "] " << vektorBeast.at(i)->getNazev() << " - "<<
      vektorBeast.at(i)->getPopis() << std::endl;
00058
00059
00060                     for(unsigned int j=0; j<vektorHumanoid.size(); index++, j++)
00061                         std::cout << "[" << index << "] " << vektorHumanoid.at(j)->getNazev() << " - "<<
      vektorHumanoid.at(j)->getPopis() << std::endl;
00062
00063                     std::cout << "Vyber si cil: ";
00064
00065                     while(true){
00067                         std::cin >> volbaCislo;std::cin.ignore();
00068
00069                         if(volbaCislo < index){
00070                             if(volbaCislo<vektorBeast.size()) boj(scrat,vektorBeast.at(volbaCislo));
00071                             else boj(scrat, vektorHumanoid.at(volbaCislo-vektorBeast.size()));
00072                             break;
00073                         }
00074                     }
00075                     index=0;
00076                     break;
00077                 default:
00078                     break;
00079             }
00080         }
00081 }
```

7.61 Dokumentace souboru cpp/ZapadniPlanina.cpp

```
#include "../hlavicky/ZapadniPlanina.h"
```

7.62 ZapadniPlanina.cpp

```
00001
00004 #include "../hlavicky/ZapadniPlanina.h"
00005
00009 IceAge::ZapadniPlanina::ZapadniPlanina(std::string novyNazev):
      Command(novyNazev) {
00010 }
00011
00015 IceAge::ZapadniPlanina::~ZapadniPlanina() {
00016 }
00017
00021 void IceAge::ZapadniPlanina::prozkoumat(
```

```
      IceAge::Veverka* scrat, std::vector<IceAge::Beast*> vektorBeast,
      std::vector<IceAge::Humanoid*> vektorHumanoid, IceAge::Reka* reka) {
00022
00023     std::cout << "Vidis mensi melu. Je tu vymena nazoru mezi [";
00024     for(unsigned int i=0;i<(vektorBeast.size()-1);i++) std::cout << vektorBeast.at(i)->getNazev() << " ";
00025     std::cout << "] a [";
00026     for(unsigned int i=0;i<vektorHumanoid.size();i++) std::cout << vektorHumanoid.at(i)->getNazev() << " ";
00027     std::cout << " ]";
00028     std::cout << std::endl;
00029     std::cout << "Radeji se drzis stranou." <<std::endl;
00030     std::cout << std::endl;
00031
00032 }
```

7.63 Dokumentace souboru cpp/Zbran.cpp

```
#include "../hlavicky/Zbran.h"
```

7.64 Zbran.cpp

```
00001
00004 #include "../hlavicky/Zbran.h"
00005
00009 short IceAge::Zbran::getPoskozeni() {
00010     return this->m_poskozeni;
00011 }
00012
00016 void IceAge::Zbran::setPoskozeni(short novePoskozeni) {
00017     if(novePoskozeni>0) this->m_poskozeni=novePoskozeni;
00018     else this->m_poskozeni=0;
00019 }
00020
00024 IceAge::Zbran::Zbran(std::string novyNazev, short novePoskozeni):
      Predmet(novyNazev) {
00025     this->setPoskozeni(novePoskozeni);
00026     this->setTyp('w');
00027 }
00028
00032 IceAge::Zbran::~Zbran(){
00033 }
00034
00038 short IceAge::Zbran::getAtribut(){
00039      return this->getPoskozeni();
00040 }
```

7.65 Dokumentace souboru cpp/Zivot.cpp

```
#include "../hlavicky/Zivot.h"
```

7.66 Zivot.cpp

```
00001
00004 #include "../hlavicky/Zivot.h"
00005
00010 void IceAge::Zivot::changeZivotAktualni(short hodnotaZmeny) {
00011     this->m_zivotAktualni+=hodnotaZmeny;
00012     if(this->m_zivotAktualni>this->m_zivotMaximalni) this->
      m_zivotAktualni=this->m_zivotMaximalni;
00016     if(this->m_zivotAktualni <= 0){
00017         std::cout << "Tohle ma byt mirumilovna hra! Nepovedlo se ti ziskat orisek! Game Over!" << std::endl
      ;
```

```
00018
00019         IceAge::MemoryArbiter::cycleDelete();
00020         exit(0);
00021     }
00022 }
00023
00027 short IceAge::Zivot::getZivotAktualni() {
00028     return this->m_zivotAktualni;
00029 }
00030
00034 short IceAge::Zivot::getZivotMaximalni() {
00035     return this->m_zivotMaximalni;
00036 }
00037
00042 void IceAge::Zivot::setZivotAktualni(short novyZivotAktualni) {
00043     if(novyZivotAktualni>=0)
00044     {
00045         if(novyZivotAktualni<this->m_zivotMaximalni) this->
      m_zivotAktualni=novyZivotAktualni;
00046         else this->m_zivotAktualni=this->m_zivotMaximalni;
00047     }
00048     else this->m_zivotAktualni=1;
00049 }
00050
00054 void IceAge::Zivot::setZivotMaximalni(short novyZivotMaximalni) {
00055     if(novyZivotMaximalni>0) this->m_zivotMaximalni=novyZivotMaximalni;
00056     else this->m_zivotMaximalni=5;
00057 }
00058
00062 IceAge::Zivot::Zivot(short novyZivot, short novyMaximalniZivot) {
00063 //nejdriv udelat maximalku, potom aktualni
00064     this->setZivotMaximalni(novyMaximalniZivot);
00065     this->setZivotAktualni(novyZivot);
00066 }
00067
00071 IceAge::Zivot::~Zivot(){
00072 }
```

7.67 Dokumentace souboru hlavicky/Beast.h

```
#include "Memory.h"
#include "Zivot.h"
#include "Nazev.h"
#include "Popis.h"
#include "Staty.h"
```

Třídy

- class IceAge::Beast

Prostory jmen

- IceAge

7.68 Beast.h

```
00001
00004 #ifndef BEAST_H
00005 #define BEAST_H
00006
00007 #include "Memory.h"
00008 #include "Zivot.h"
00009 #include "Nazev.h"
00010 #include "Popis.h"
```

```
00011 #include "Staty.h"
00012
00013 namespace IceAge {
00017     class Beast : public IceAge::Memory, public IceAge::Zivot, public
      IceAge::Nazev, public IceAge::Popis, public
      IceAge::Staty {
00018
00019     private:
00020         short m_zurivost;
00021         void setZurivost(short novaZurivost);
00023     public:
00024         short utok();
00025         short getZurivost();
00026         short obrana();
00027         Beast(short novaZurivost, std::string novyNazev, short novaSila, std::string novyPopis, short
      novaOdolnost, short novyZivot);
00028         ~Beast();
00029     };
00030 }
00031 #endif
```

7.69 Dokumentace souboru hlavicky/BuilderBytost.h

```
#include "BuilderPredmet.h"
#include "Beast.h"
#include "Humanoid.h"
```

Třídy

- class IceAge::BuilderBytost

Prostory jmen

- IceAge

7.70 BuilderBytost.h

```
00001
00004 #ifndef BUILDERBYTOST_H
00005 #define BUILDERBYTOST_H
00006
00007 #include "BuilderPredmet.h"
00008 #include "Beast.h"
00009 #include "Humanoid.h"
00010
00011 namespace IceAge {
00012
00016     class BuilderBytost {
00017
00018     private:
00019         IceAge::Beast* m_beast;
00020         IceAge::Humanoid* m_humanoid;
00021         IceAge::BuilderPredmet *m_builder;
00023     public:
00024         BuilderBytost(IceAge::BuilderPredmet* novyBuilder);
00025         ~BuilderBytost();
00027         void setBuilder(IceAge::BuilderPredmet* novyBuilder);
00029         IceAge::Predmet* createPredmet(std::string novyNazev, short novaHodnota
      );
00031         IceAge::Beast* getBeast();
00032         IceAge::Humanoid* getHumanoid();
00034         void createBeast(short novaZurivost, std::string novyNazev, short novaSila, std::string
      novyPopis, short novaOdolnost, short novyZivot);
00036         void createHumanoid(short novyZivot, std::string novyPopis, short novaSila, short
      novaOdolnost, std::string novyNazevPredmetu1, std::string novyNazevPredmetu2,
00037                                                 std::string novyNazevPredmetu3, short novaHodnotaPredmetu1,
      short novaHodnotaPredmetu2, short novaHodnotaPredmetu3);
00038     };
00039 }
00040 #endif
```

7.71 Dokumentace souboru hlavicky/BuilderLokace.h

```
#include "BuilderBytost.h"
#include "Lokace.h"
#include <iostream>
#include <fstream>
```

Třídy

- class IceAge::BuilderLokace

Prostory jmen

- IceAge

7.72 BuilderLokace.h

```
00001
00004 #ifndef BUILDERLOKACE_H
00005 #define BUILDERLOKACE_H
00006
00007 #include "BuilderBytost.h"
00008 #include "Lokace.h"
00009 #include <iostream>
00010
00011 #include <fstream>
00012
00013
00014 namespace IceAge {
00018     class BuilderLokace {
00019
00020     private:
00021         IceAge::BuilderBytost* m_builderBytost;
00022         IceAge::Lokace* m_lokace;
00023         std::vector <IceAge::Lokace*> m_vektorLokaci;
00024     public:
00025         BuilderLokace(IceAge::BuilderBytost* novyBuilder);
00026         ~BuilderLokace();
00028         void setBuilder(IceAge::BuilderBytost* novyBuilder);
00030         void createLokace();
00032         IceAge::Lokace* getLokace();
00034         IceAge::Humanoid* createHumanoid(short novyZivot, std::string
      novyPopis, short novaSila, short novaOdolnost,
00035                                          std::string novyNazevPredmetu1, std::string novyNazevPredmetu2,
      std::string novyNazevPredmetu3,
00036                                          short novaHodnotaPredmetu1, short novaHodnotaPredmetu2, short
      novaHodnotaPredmetu3);
00038         IceAge::Beast* createBeast(short novaZurivost, std::string novyNazev, short
       novaSila, std::string novyPopis, short novaOdolnost, short novyZivot);
00040         void nactiBeasty(std::ifstream &popisLokace);
00041         void nactiHumanoidy(std::ifstream &popisLokace);
00042     };
00043 }
00044 #endif
```

7.73 Dokumentace souboru hlavicky/BuilderPredmet.h

```
#include "Zbran.h"
```

Třídy

- class IceAge::BuilderPredmet

Prostory jmen

- IceAge

7.74 BuilderPredmet.h

```
00001
00004 #ifndef BUILDERPREDMET_H
00005 #define BUILDERPREDMET_H
00006
00007 #include "Zbran.h"
00008
00009 namespace IceAge {
00010
00014     class BuilderPredmet {
00015
00016     private:
00017         IceAge::Predmet* m_pripravovany;
00019     public:
00020         void createPredmet(std::string novyNazev, short novaHodnota);
00022         IceAge::Predmet* getPredmet();
00024         BuilderPredmet();
00025         ~BuilderPredmet();
00026     };
00027 }
00028
00029 #endif
```

7.75 Dokumentace souboru hlavicky/Command.h

```
#include <iostream>
#include "Nazev.h"
#include "Veverka.h"
#include "Reka.h"
#include "Beast.h"
#include "Humanoid.h"
#include "VolneFunkce.h"
```

Třídy

- class IceAge::Command

Prostory jmen

- IceAge

7.76 Command.h

```
00001
00004 #ifndef COMMAND_H
00005 #define COMMAND_H
00006
00007
00008 #include <iostream>
00009 #include "Nazev.h"
00010 #include "Veverka.h"
00011 #include "Reka.h"
00012 #include "Beast.h"
00013 #include "Humanoid.h"
00014 #include "VolneFunkce.h"
00015
00016 namespace IceAge {
00020     class Command : public IceAge::Nazev {
00021
00022     public:
00023         Command(std::string novyNazev);
00025         virtual ~Command();
00027         virtual void prozkoumat(IceAge::Veverka* scrat,
      std::vector<IceAge::Beast*> vektorBeast,
00028                                 std::vector<IceAge::Humanoid*> vektorHumanoid,
      IceAge::Reka* reka) = 0;
00029     };
00030 }
00031
00032 #endif
```

7.77 Dokumentace souboru hlavicky/EngineIceAge.h

```
#include "LocationDirector.h"
#include "Veverka.h"
#include "Hra.h"
```

Třídy

- class IceAge::EngineIceAge

Prostory jmen

- IceAge

7.78 EngineIceAge.h

```
00001
00004 #ifndef ENGINEICEAGE_H
00005 #define ENGINEICEAGE_H
00006
00007 #include "LocationDirector.h"
00008 #include "Veverka.h"
00009 #include "Hra.h"
00010
00011
00012 namespace IceAge {
00017     class EngineIceAge : public IceAge::Memory {
00018
00019     private:
00020         IceAge::Veverka* m_veverka;
00021         static IceAge::EngineIceAge* s_vlastniInstance;
00022         IceAge::LocationDirector* m_directorLokaci;
00023         IceAge::Lokace* m_lokace;
```

```
00025         EngineIceAge(IceAge::LocationDirector *novyDirector);
00027         IceAge::Veverka* getVeverka();
00029         void createVeverka();
00030         bool inicializaceLokaci();
00032         void setDirector(IceAge::LocationDirector* novyDirector);
00034     public:
00035         static IceAge::EngineIceAge* getInstance();
00036         void startHry();
00038         ~EngineIceAge();
00039     };
00040 }
00041 #endif
```

7.79 Dokumentace souboru hlavicky/Hra.h

```
#include "Lokace.h"
#include "Veverka.h"
#include "VolneFunkce.h"
```

Třídy

- class IceAge::Hra

Prostory jmen

- IceAge

7.80 Hra.h

```
00001
00004 #ifndef HRA_H
00005 #define HRA_H
00006
00007 #include "Lokace.h"
00008 #include "Veverka.h"
00009 #include "VolneFunkce.h"
00010
00011 namespace IceAge {
00015     class Hra : public IceAge::Memory {
00016
00017     private:
00018         IceAge::Lokace* m_aktualniLokace;
00019         short m_obrazek;
00021         void zmenLokaci(IceAge::Veverka* scrat);
00022         void zmenLokaci(unsigned int index, IceAge::Veverka* scrat);
00024         void setAktualniLokace(IceAge::Lokace* novaAktualniLokace);
00025         void vypisAktualniLokaci();
00026         void vypisAsciObrazek();
00027         void intro(IceAge::Veverka *scrat);
00029     public:
00030         Hra(IceAge::Lokace* novaAktualniLokace);
00031         void zacniHrat(IceAge::Veverka *scrat);
00033     };
00034 }
00035 #endif
```

7.81 Dokumentace souboru hlavicky/Humanoid.h

```
#include "Inventar.h"
#include "ObleceneVybaveniHumanoid.h"
#include "Memory.h"
#include "Zivot.h"
#include "Nazev.h"
#include "Popis.h"
#include "Staty.h"
```

Třídy

- class IceAge::Humanoid

Prostory jmen

- IceAge

7.82 Humanoid.h

```
00001
00004 #ifndef HUMANOID_H
00005 #define HUMANOID_H
00006
00007 #include "Inventar.h"
00008 #include "ObleceneVybaveniHumanoid.h"
00009 #include "Memory.h"
00010 #include "Zivot.h"
00011 #include "Nazev.h"
00012 #include "Popis.h"
00013 #include "Staty.h"
00014
00015 namespace IceAge {
00019     class Humanoid : public IceAge::Inventar, public
      IceAge::ObleceneVybaveniHumanoid, public
      IceAge::Memory,
00020                      public IceAge::Zivot, public IceAge::Nazev, public
      IceAge::Popis, public IceAge::Staty {
00021
00022     public:
00023         Humanoid(short novyZivot, std::string novyNazev, std::string novyPopis, short novaSila,
      short novaOdolnost, IceAge::Predmet* zbran1, IceAge::Predmet* zbran2,
      IceAge::Predmet* zbran3);
00024         ~Humanoid();
00026         short utok();
00028         short obrana();
00029     };
00030 }
00031
00032 #endif
```

7.83 Dokumentace souboru hlavicky/IInventar.h

```
#include "Predmet.h"
```

Třídy

- class IceAge::IInventar

Prostory jmen

- IceAge

7.84 IInventar.h

```
00001
00004 #ifndef IINVENTAR_H
00005 #define IINVENTAR_H
00006
00007 #include"Predmet.h"
00008
00009 namespace IceAge {
00013     class IInventar {
00014
00015     protected:
00016         virtual unsigned short getMaximalniVelikost() = 0;
00017         virtual void setMaximalniVelikost(unsigned short novaMaximalniVelikost) = 0;
00019     public:
00020         virtual void pridejPredmet(IceAge::Predmet* novyPredmet) = 0;
00021         virtual IceAge::Predmet* odeberPredmet() = 0;
00022         virtual void znicPredmet() = 0;
00023     };
00024 }
00025 #endif
```

7.85 Dokumentace souboru hlavicky/IMemory.h

Třídy

- class IceAge::IMemory

Prostory jmen

- IceAge

7.86 IMemory.h

```
00001
00004 #ifndef IMEMORY_H
00005 #define IMEMORY_H
00006
00007
00008
00009 namespace IceAge {
00013     class IMemory {
00014
00015     protected:
00016         virtual unsigned int getId() = 0;
00018     public:
00019         virtual void setId(unsigned int novaId) = 0;
00020     };
00021 }
00022 #endif
```

7.87 Dokumentace souboru hlavicky/INazev.h

```
#include <iostream>
```

Třídy

- class IceAge::INazev

Prostory jmen

- IceAge

7.88 INazev.h

```
00001
00004 #ifndef INAZEV_H
00005 #define INAZEV_H
00006
00007 #include<iostream>
00008
00009 namespace IceAge {
00013     class INazev {
00014
00015     protected:
00016         virtual void setNazev(std::string novyNazev) = 0;
00018     public:
00019         virtual std::string getNazev() = 0;
00020     };
00021 }
00022 #endif
```

7.89 Dokumentace souboru hlavicky/Inventar.h

```
#include "IInventar.h"
```

Třídy

- class IceAge::Inventar

Prostory jmen

- IceAge

Proměnné

- const short MAX_INVENTAR =5

  Konstanta udavajici maximalni velikost inventare.

7.89.1 Dokumentace proměnných

7.89.1.1 const short MAX_INVENTAR =5

Konstanta udavajici maximalni velikost inventare.

Definice je uvedena na řádku 8 v souboru Inventar.h.

7.90 Inventar.h

```
00001
00004 #ifndef INVENTAR_H
00005 #define INVENTAR_H
00006
00007 #include"IInventar.h"
00008 const short MAX_INVENTAR=5;
00010 namespace IceAge {
00014     class Inventar : public IceAge::IInventar {
00015
00016     protected:
00017         std::vector<IceAge::Predmet*> m_vektorPredmetu;
00018         unsigned short m_maximalniVelikost;
00020         unsigned short getMaximalniVelikost();
00021         void setMaximalniVelikost(unsigned short novaMaximalniVelikost);
00022         unsigned int volba();
00024     public:
00025         IceAge::Predmet* odeberPredmet();
00026         void pridejPredmet(IceAge::Predmet* novyPredmet);
00027         void znicPredmet();
00028         void vypisObsah();
00030         Inventar(unsigned short novaVelikost);
00031         ~Inventar();
00032     };
00033 }
00034 #endif
```

7.91 Dokumentace souboru hlavicky/IObleceneVybaveni.h

```
#include "Zbran.h"
```

Třídy

- class IceAge::IObleceneVybaveni

Prostory jmen

- IceAge

7.92 IObleceneVybaveni.h

```
00001
00004 #ifndef IOBLECENEVYBAVENI_H
00005 #define IOBLECENEVYBAVENI_H
00006
00007 #include"Zbran.h"
00008
00009 namespace IceAge {
00013     class IObleceneVybaveni {
00014
00015     protected:
00016         virtual IceAge::Predmet* getLevaRuka() = 0;
00017         virtual IceAge::Predmet* getPravaRuka() = 0;
00019     public:
00020         virtual void setLevaRuka(IceAge::Predmet* novaLevaRuka) = 0;
00021         virtual void setPravaRuka(IceAge::Predmet* novaPravaRuka) = 0;
00022     };
00023 }
00024 #endif
```

7.93 Dokumentace souboru hlavicky/IPopis.h

```
#include <iostream>
```

Třídy

- class IceAge::IPopis

Prostory jmen

- IceAge

7.94 IPopis.h

```
00001
00004 #ifndef IPOPIS_H
00005 #define IPOPIS_H
00006
00007 #include<iostream>
00008
00009 namespace IceAge {
00013     class IPopis {
00014
00015     protected:
00016         virtual std::string getPopis() = 0;
00018         virtual void setPopis(std::string novyPopis) = 0;
00019     };
00020 }
00021 #endif
```

7.95 Dokumentace souboru hlavicky/IStaty.h

Třídy

- class IceAge::IStaty

Prostory jmen

- IceAge

7.96 IStaty.h

```
00001
00004 #ifndef ISTATY_H
00005 #define ISTATY_H
00006
00007 namespace IceAge {
00011     class IStaty {
00012
00013     protected:
00014         virtual short getSila() = 0;
00016         virtual void setSila(short novaSila) = 0;
00018         virtual short getOdolnost() = 0;
00020         virtual void setOdolnost(short novaOdolnost) = 0;
00021     };
00022 }
00023 #endif
```

7.97 Dokumentace souboru hlavicky/IZivot.h

```
#include <iostream>
```

Třídy

- class IceAge::IZivot

Prostory jmen

- IceAge

7.98 IZivot.h

```
00001
00004 #ifndef IZIVOT_H
00005 #define IZIVOT_H
00006
00007 #include<iostream>
00008
00009
00010 namespace IceAge {
00014     class IZivot {
00015
00016
00017     protected:
00018         virtual short getZivotAktualni() = 0;
00020         virtual void setZivotAktualni(short novyZivotAktualni) = 0;
00022         virtual short getZivotMaximalni() = 0;
00024         virtual void setZivotMaximalni(short novyZivotMaximalni) = 0;
00026     public:
00027         virtual void changeZivotAktualni(short hodnotaZmeny) = 0;
00029     };
00030 }
00031 #endif
```

7.99 Dokumentace souboru hlavicky/Jezero.h

```
#include "Command.h"
#include <thread>
#include <chrono>
```

Třídy

- class IceAge::Jezero

Prostory jmen

- IceAge

7.100 Jezero.h

```
00001
00004 #ifndef JEZERO_H
00005 #define JEZERO_H
00006
00007 #include "Command.h"
00008
00009 #include <thread>
00010 #include <chrono>
00011
00012 namespace IceAge {
00016     class Jezero : public IceAge::Command {
00017
00018
00019     public:
00020         Jezero(std::string novyNazev);
00022         ~Jezero();
00024         void prozkoumat(IceAge::Veverka* scrat, std::vector<IceAge::Beast*>
      vektorBeast, std::vector<IceAge::Humanoid*> vektorHumanoid, IceAge::Reka* reka);
00025     };
00026 }
00027 #endif
```

7.101 Dokumentace souboru hlavicky/JizniLes.h

```
#include "Command.h"
```

Třídy

- class IceAge::JizniLes

Prostory jmen

- IceAge

7.102 JizniLes.h

```
00001
00004 #ifndef JIZNILES_H
00005 #define JIZNILES_H
00006
00007 #include "Command.h"
00008
00009 namespace IceAge {
00013     class JizniLes : public IceAge::Command {
00014
00015
00016     public:
00017         JizniLes(std::string novyNazev);
00019         ~JizniLes();
00021         void prozkoumat(IceAge::Veverka* scrat, std::vector<IceAge::Beast*>
      vektorBeast, std::vector<IceAge::Humanoid*> vektorHumanoid, IceAge::Reka* reka);
00022     };
00023 }
00024 #endif
```

7.103 Dokumentace souboru hlavicky/Ledovec.h

```
#include "Command.h"
```

Třídy

- class IceAge::Ledovec

Prostory jmen

- IceAge

7.104 Ledovec.h

```
00001
00004 #ifndef LEDOVEC_H
00005 #define LEDOVEC_H
00006 #include "Command.h"
00007
00008
00009
00010 namespace IceAge {
00014     class Ledovec : public IceAge::Command {
00015
00016
00017     public:
00018         Ledovec(std::string novyNazev);
00020         ~Ledovec();
00022         void prozkoumat(IceAge::Veverka* scrat, std::vector<IceAge::Beast*>
      vektorBeast, std::vector<IceAge::Humanoid*> vektorHumanoid, IceAge::Reka* reka);
00023     };
00024 }
00025 #endif
```

7.105 Dokumentace souboru hlavicky/LocationDirector.h

```
#include "BuilderLokace.h"
```

Třídy

- class IceAge::LocationDirector

Prostory jmen

- IceAge

7.106 LocationDirector.h

```
00001
00004 #ifndef LOCATIONDIRECTOR_H
00005 #define LOCATIONDIRECTOR_H
00006
00007 #include"BuilderLokace.h"
00008
00009 namespace IceAge {
00013     class LocationDirector {
00014
00015     private:
00016         IceAge::BuilderLokace* m_builder;
00018     public:
00019         LocationDirector(IceAge::BuilderLokace* novyBuilder);
00020         ~LocationDirector();
00022         IceAge::Lokace* createLokace();
00024         void setBuilder(IceAge::BuilderLokace* novyBuilder);
00025     };
00026 }
00027
00028 #endif
```

7.107 Dokumentace souboru hlavicky/Lokace.h

```
#include "Command.h"
#include "Reka.h"
#include "Popis.h"
#include "Nazev.h"
#include "Memory.h"
#include <vector>
#include "Zivot.h"
#include "Veverka.h"
#include "Beast.h"
#include "Humanoid.h"
#include "Ledovec.h"
#include "ZapadniPlanina.h"
#include "ZapadniBrehReky.h"
#include "JizniLes.h"
#include "SeverniLes.h"
#include "VychodniPlanina.h"
#include "VychodniBrehReky.h"
#include "Jezero.h"
```

Třídy

- class IceAge::Lokace

Prostory jmen

- IceAge

Proměnné

- const short LECIVOST_REKY =5

7.107.1 Dokumentace proměnných

7.107.1.1 const short LECIVOST_REKY =5

Definice je uvedena na řádku 30 v souboru Lokace.h.

7.108 Lokace.h

```
00001
00004 #ifndef LOKACE_H
00005 #define LOKACE_H
00006
00007 namespace IceAge{
00008     class Command;
00009 }
00010 #include "Command.h"
00011 #include "Reka.h"
00012 #include "Popis.h"
00013 #include "Nazev.h"
00014 #include "Memory.h"
00015 #include <vector>
00016 #include "Zivot.h"
00017 #include "Veverka.h"
00018 #include "Beast.h"
00019 #include "Humanoid.h"
00020
00021 #include "Ledovec.h"
00022 #include "ZapadniPlanina.h"
00023 #include "ZapadniBrehReky.h"
00024 #include "JizniLes.h"
00025 #include "SeverniLes.h"
00026 #include "VychodniPlanina.h"
00027 #include "VychodniBrehReky.h"
00028 #include "Jezero.h"
00029
00030 const short LECIVOST_REKY=5;
00031
00032 namespace IceAge {
00033
00034     class Lokace : public IceAge::Nazev, public IceAge::Popis, public
      IceAge::Memory {
00035
00036     private:
00037         std::vector<IceAge::Beast*> m_vektorBeast;
00038         std::vector<IceAge::Humanoid*> m_vektorHumanoid;
00039         IceAge::Reka* m_odkazReka;
00040         std::vector<IceAge::Lokace*> m_vektorLokaci;
00041         IceAge::Command* m_prikaz;
00042
00043     public:
00044         Lokace(std::string novyNazev, std::string novyPopis);
00045         ~Lokace();
00046
00047         void pridejBeast(IceAge::Beast* novyBeast);
00048         void pridejHumanoid(IceAge::Humanoid* novyHumanoid);
00049
00050         void smazBeast(unsigned int index);
00051         void smazHumanoid(unsigned int index);
00052
00053         void pridejReku();
00054
00055         void smazReku();
```

```
00056
00057         void pridejLokaci(IceAge::Lokace* novaLokace);
00058
00059         void smazLokaci(unsigned int index);
00060
00061         void odeberLokaci(unsigned int index);
00062
00063         void vypisLokace();
00064
00065         std::string getNazevLokace();
00066         std::string getPopisLokace();
00067         Lokace* getLokace(unsigned int index);
00068         IceAge::Beast* getBeast(unsigned int index);
00069         IceAge::Humanoid* getHumanoid(unsigned int index);
00070
00071         unsigned int getVelikostVektoruLokaci();
00072
00073         std::vector<IceAge::Beast*> getVektorBeast();
00074         std::vector<IceAge::Humanoid*> getVektorHumanoid();
00075         IceAge::Reka* getReka();
00076
00077         void setPrikaz();
00078
00079         void prozkoumatLokaci(IceAge::Veverka* scrat);
00080
00081         //unsigned int volba();
00082     };
00083 }
00084 #endif
```

7.109 Dokumentace souboru hlavicky/Memory.h

```
#include "IMemory.h"
#include "MemoryArbiter.h"
```

Třídy

- class IceAge::Memory

Prostory jmen

- IceAge

7.110 Memory.h

```
00001
00004 #ifndef MEMORY_H
00005 #define MEMORY_H
00006
00007 namespace IceAge{
00008     class MemoryArbiter;
00009 }
00010
00011 #include "IMemory.h"
00012 #include "MemoryArbiter.h"
00013
00014
00015
00016 namespace IceAge {
00020     class Memory : public IceAge::IMemory {
00021
00022     protected:
00023         unsigned int m_id;
00025         unsigned int getId();
00027     public:
00028         void setId(unsigned int noveId);
00030         Memory();
00031         virtual ~Memory();
00032     };
00033 }
00034 #endif
```

7.111 Dokumentace souboru hlavicky/MemoryArbiter.h

```
#include "Memory.h"
#include <vector>
#include <iostream>
```

Třídy

- class IceAge::MemoryArbiter

Prostory jmen

- IceAge

7.112 MemoryArbiter.h

```
00001
00004 #ifndef MEMORYARBITER_H
00005 #define MEMORYARBITER_H
00006
00007 namespace IceAge{
00008     class Memory;
00009 }
00010 #include "Memory.h"
00011 #include <vector>
00012 #include <iostream>
00013
00014
00015 namespace IceAge {
00022     class MemoryArbiter {
00023
00024     private:
00025         static std::vector<IceAge::Memory*> s_memoryVector;
00027         MemoryArbiter();
00029     public:
00030         static void add(IceAge::Memory* target);
00032         static void remove(unsigned int targetId);
00034         static void cycleDelete();
00035     };
00036 }
00037 #endif
```

7.113 Dokumentace souboru hlavicky/Nazev.h

```
#include "INazev.h"
```

Třídy

- class IceAge::Nazev

Prostory jmen

- IceAge

7.114 Nazev.h

```
00001
00004 #ifndef NAZEV_H
00005 #define NAZEV_H
00006
00007 #include"INazev.h"
00008
00009
00010 namespace IceAge {
00014     class Nazev : public IceAge::INazev {
00015
00016     protected:
00017         std::string m_nazev;
00019         void setNazev(std::string novyNazev);
00021     public:
00022         Nazev(std::string novyNazev);
00023         std::string getNazev();
00024     };
00025 }
00026 #endif
```

7.115 Dokumentace souboru hlavicky/ObleceneVybaveniHumanoid.h

```
#include "IObleceneVybaveni.h"
```

Třídy

- class IceAge::ObleceneVybaveniHumanoid

Prostory jmen

- IceAge

7.116 ObleceneVybaveniHumanoid.h

```
00001
00004 #ifndef OBLECENEVYBAVENIHUMANOID_H
00005 #define OBLECENEVYBAVENIHUMANOID_H
00006
00007 #include"IObleceneVybaveni.h"
00008
00009
00010 namespace IceAge {
00014     class ObleceneVybaveniHumanoid : public
      IceAge::IObleceneVybaveni {
00015
00016     protected:
00017         IceAge::Predmet* m_levaRuka;
00018         IceAge::Predmet* m_pravaRuka;
00019         IceAge::Predmet* m_zada;
00021         IceAge::Predmet* getLevaRuka();
00022         IceAge::Predmet* getPravaRuka();
00023         IceAge::Predmet* getZada();
00025     public:
00026         void setZada(IceAge::Predmet* novaZada);
00027         void setPravaRuka(IceAge::Predmet* novaPravaRuka);
00028         void setLevaRuka(IceAge::Predmet* novaLevaRuka);
00029         ObleceneVybaveniHumanoid(IceAge::Predmet* levaRuka,
      IceAge::Predmet* pravaRuka, IceAge::Predmet* zada);
00030         ~ObleceneVybaveniHumanoid();
00031     };
00032 }
00033 #endif
```

7.117 Dokumentace souboru hlavicky/ObleceneVybaveniVeverky.h

```
#include "IObleceneVybaveni.h"
#include "Predmet.h"
#include "Zbran.h"
#include "Orisek.h"
```

Třídy

- class IceAge::ObleceneVybaveniVeverky

Prostory jmen

- IceAge

7.118 ObleceneVybaveniVeverky.h

```
00001
00004 #ifndef OBLECENEVYBAVENIVEVERKY_H
00005 #define OBLECENEVYBAVENIVEVERKY_H
00006
00007 #include"IObleceneVybaveni.h"
00008 #include"Predmet.h"
00009 #include "Zbran.h"
00010 #include "Orisek.h"
00011
00012 namespace IceAge {
00016     class ObleceneVybaveniVeverky : public
      IceAge::IObleceneVybaveni {
00017
00018     protected:
00019         IceAge::Predmet* m_levaRuka;
00020         IceAge::Predmet* m_pravaRuka;
00022     public:
00023         void setPravaRuka(IceAge::Predmet* novaPravaRuka);
00024         void setLevaRuka(IceAge::Predmet* novaLevaRuka);
00025         ObleceneVybaveniVeverky(IceAge::Predmet* novaLevaRuka,
      IceAge::Predmet* novaPravaRuka);
00026         ~ObleceneVybaveniVeverky();
00027         IceAge::Predmet* getLevaRuka();
00028         IceAge::Predmet* getPravaRuka();
00030         void zahodLevaRuka();
00031         void zahodPravaRuka();
00032     };
00033 }
00034 #endif
```

7.119 Dokumentace souboru hlavicky/Orisek.h

```
#include "Predmet.h"
```

Třídy

- class IceAge::Orisek

Prostory jmen

- IceAge

7.120 Orisek.h

```
00001
00004 #ifndef ORISEK_H
00005 #define ORISEK_H
00006
00007 #include"Predmet.h"
00008 namespace IceAge {
00013     class Orisek : public IceAge::Predmet {
00014
00015     public:
00016         Orisek(std::string novyPopis);
00017         ~Orisek();
00018          short getAtribut();
00019     };
00020 }
00021 #endif
```

7.121 Dokumentace souboru hlavicky/Popis.h

```
#include "IPopis.h"
```

Třídy

- class IceAge::Popis

Prostory jmen

- IceAge

7.122 Popis.h

```
00001
00004 #ifndef POPIS_H
00005 #define POPIS_H
00006
00007 #include"IPopis.h"
00008
00009 namespace IceAge {
00013     class Popis : public IceAge::IPopis {
00014
00015     protected:
00016         std::string m_popis;
00018         void setPopis(std::string novyPopis);
00020     public:
00021         Popis(std::string novyPopis);
00022         ~Popis();
00023         std::string getPopis();
00024     };
00025 }
00026 #endif
```

7.123 Dokumentace souboru hlavicky/Predmet.h

```
#include "Memory.h"
#include "Nazev.h"
```

Třídy

- class IceAge::Predmet

Prostory jmen

- IceAge

7.124 Predmet.h

```
00001
00004 #ifndef PREDMET_H
00005 #define PREDMET_H
00006
00007 #include"Memory.h"
00008 #include"Nazev.h"
00009
00010
00011 namespace IceAge {
00015     class Predmet : public IceAge::Memory, public
      IceAge::Nazev {
00016
00017     protected:
00018         char m_typ;
00020         void setTyp(char novyTyp);
00023     public:
00024         char getTyp();
00026         virtual short getAtribut() = 0;
00027         Predmet(std::string novyNazev);
00028         ~Predmet();
00029     };
00030 }
00031 #endif
```

7.125 Dokumentace souboru hlavicky/Reka.h

Třídy

- class IceAge::Reka

Prostory jmen

- IceAge

7.126 Reka.h

```
00001
00004 #ifndef REKA_H
00005 #define REKA_H
00006
00007 namespace IceAge {
00011     class Reka {
00012
00013     private:
00014         short m_hodnotaZivota;
00016         void setHodnotaZivota(short novaHodnotaZivota);
00018     public:
00019         short getHodnotaZivota();
00021         Reka(short novaHodnota);
00022         ~Reka();
00023     };
00024 }
00025 #endif
```

7.127 Dokumentace souboru hlavicky/SeverniLes.h

```
#include "Command.h"
```

Třídy

- class IceAge::SeverniLes

Prostory jmen

- IceAge

7.128 SeverniLes.h

```
00001
00004 #ifndef SEVERNILES_H
00005 #define SEVERNILES_H
00006
00007 #include "Command.h"
00008
00009 namespace IceAge {
00013     class SeverniLes : public IceAge::Command {
00014
00015
00016     public:
00017         SeverniLes(std::string novyNazev);
00019         ~SeverniLes();
00021         void prozkoumat(IceAge::Veverka* scrat, std::vector<IceAge::Beast*>
      vektorBeast, std::vector<IceAge::Humanoid*> vektorHumanoid, IceAge::Reka* reka);
00022     };
00023 }
00024 #endif
```

7.129 Dokumentace souboru hlavicky/Staty.h

```
#include "IStaty.h"
#include <iostream>
```

Třídy

- class IceAge::Staty

Prostory jmen

- IceAge

7.130 Staty.h

```
00001
00004 #ifndef STATY_H
00005 #define STATY_H
00006
00007 #include"IStaty.h"
00008 #include <iostream>
00009
00010 namespace IceAge {
00014     class Staty : public IceAge::IStaty {
00015
00016     protected:
00017         short m_sila;
00018         short m_odolnost;
00020         short getSila();
00021         void setSila(short novaSila);
00023         short getOdolnost();
00024         void setOdolnost(short novaOdolnost);
00026     public:
00027         Staty(short novaSila, short novaOdolnost);
00028         ~Staty();
00029     };
00030 }
00031 #endif
```

7.131 Dokumentace souboru hlavicky/Veverka.h

```
#include "ObleceneVybaveniVeverky.h"
#include "Staty.h"
#include "Memory.h"
#include "Zivot.h"
#include <cstdio>
```

Třídy

- class IceAge::Veverka

Prostory jmen

- IceAge

Proměnné

- const short SCRAT_HP =20

  Konstanta urcujici hracovo zdravi.
- const short SCRAT_SILA =5

  Konstanta urcujici hracovu silu.
- const short SCRAT_ODOLNOST =5

  Konstanta urcujici hracovu odolnost.

7.131.1 Dokumentace proměnných

7.131.1.1 const short SCRAT_HP =20

Konstanta urcujici hracovo zdravi.

Definice je uvedena na řádku 13 v souboru Veverka.h.

7.131.1.2 const short SCRAT_ODOLNOST =5

Konstanta urcujici hracovu odolnost.

Definice je uvedena na řádku 15 v souboru Veverka.h.

7.131.1.3 const short SCRAT_SILA =5

Konstanta urcujici hracovu silu.

Definice je uvedena na řádku 14 v souboru Veverka.h.

7.132 Veverka.h

```
00001
00004 #ifndef VEVERKA_H
00005 #define VEVERKA_H
00006
00007 #include "ObleceneVybaveniVeverky.h"
00008 #include "Staty.h"
00009 #include "Memory.h"
00010 #include "Zivot.h"
00011 #include <cstdio>
00012
00013 const short SCRAT_HP=20;
00014 const short SCRAT_SILA=5;
00015 const short SCRAT_ODOLNOST=5;
00017 namespace IceAge {
00021     class Veverka : public IceAge::ObleceneVybaveniVeverky, public
      IceAge::Staty, public IceAge::Memory, public
      IceAge::Zivot {
00022
00023     private:
00024         static IceAge::Veverka* s_vlastniInstance;
00026         Veverka(short novyZivot, short novaSila, short novaOdolnost);
00028     public:
00029         static IceAge::Veverka* getInstance();
00030         short utok();
00031         short obrana();
00032         ~Veverka();
00033         void enrageZapnout();
00034         void enrageVypnout();
00035     };
00036 }
00037
00038 #endif
```

7.133 Dokumentace souboru hlavicky/VolneFunkce.h

```
#include "Zivot.h"
```

Prostory jmen

- IceAge

Funkce

- void IceAge::boj (IceAge::Zivot *utocnik, IceAge::Zivot *obrance)

7.134 VolneFunkce.h

```
00001
00004 #ifndef VOLNEFUNKCE_H
00005 #define VOLNEFUNKCE_H
00006
00007 #include"Zivot.h"
00008
00009
00010 namespace IceAge{
00014     void boj(IceAge::Zivot* utocnik, IceAge::Zivot* obrance);
00015 }
00016
00017 #endif
```

7.135 Dokumentace souboru hlavicky/VychodniBrehReky.h

```
#include "Command.h"
```

Třídy

- class IceAge::VychodniBrehReky

Prostory jmen

- IceAge

7.136 VychodniBrehReky.h

```
00001
00004 #ifndef VYCHODNIBREHREKY_H
00005 #define VYCHODNIBREHREKY_H
00006
00007 #include"Command.h"
00008
00009 namespace IceAge {
00013     class VychodniBrehReky : public IceAge::Command {
00014
00015
00016     public:
00017         VychodniBrehReky(std::string novyNazev);
00019         ~VychodniBrehReky();
00021         void prozkoumat(IceAge::Veverka* scrat, std::vector<IceAge::Beast*>
      vektorBeast, std::vector<IceAge::Humanoid*> vektorHumanoid, IceAge::Reka* reka);
00022     };
00023 }
00024
00025 #endif
```

7.137 Dokumentace souboru hlavicky/VychodniPlanina.h

```
#include "Command.h"
```

Třídy

- class IceAge::VychodniPlanina

Prostory jmen

- IceAge

7.138 VychodniPlanina.h

```
00001
00004 #ifndef VYCHODNIPLANINA_H
00005 #define VYCHODNIPLANINA_H
00006
00007 #include"Command.h"
00008
00009 namespace IceAge {
00013     class VychodniPlanina : public IceAge::Command {
00014
00015
00016     public:
00017         VychodniPlanina(std::string novyNazev);
00019         ~VychodniPlanina();
00021         void prozkoumat(IceAge::Veverka* scrat, std::vector<IceAge::Beast*>
      vektorBeast, std::vector<IceAge::Humanoid*> vektorHumanoid, IceAge::Reka* reka);
00022     };
00023 }
00024
00025 #endif
```

7.139 Dokumentace souboru hlavicky/ZapadniBrehReky.h

```
#include "Command.h"
```

Třídy

- class IceAge::ZapadniBrehReky

Prostory jmen

- IceAge

7.140 ZapadniBrehReky.h

```
00001
00004 #ifndef ZAPADNIBREHREKY_H
00005 #define ZAPADNIBREHREKY_H
00006
00007 #include "Command.h"
00008
00009
00010 namespace IceAge {
00014     class ZapadniBrehReky : public IceAge::Command {
00015
00016
00017     public:
00018         ZapadniBrehReky(std::string novyNazev);
00020         ~ZapadniBrehReky();
00022         void prozkoumat(IceAge::Veverka* scrat, std::vector<IceAge::Beast*>
      vektorBeast, std::vector<IceAge::Humanoid*> vektorHumanoid, IceAge::Reka* reka);
00023     };
00024 }
00025
00026 #endif
```

7.141 Dokumentace souboru hlavicky/ZapadniPlanina.h

```
#include "Command.h"
```

Třídy

- class IceAge::ZapadniPlanina

Prostory jmen

- IceAge

7.142 ZapadniPlanina.h

```
00001
00004 #ifndef ZAPADNIPLANINA_H
00005 #define ZAPADNIPLANINA_H
00006
00007 #include"Command.h"
00008
00009 namespace IceAge {
00013     class ZapadniPlanina : public IceAge::Command {
00014
00015
00016     public:
00017         ZapadniPlanina(std::string novyNazev);
00019         ~ZapadniPlanina();
00021         void prozkoumat(IceAge::Veverka* scrat, std::vector<IceAge::Beast*>
      vektorBeast, std::vector<IceAge::Humanoid*> vektorHumanoid, IceAge::Reka* reka);
00022     };
00023 }
00024
00025 #endif
```

7.143 Dokumentace souboru hlavicky/Zbran.h

```
#include "Predmet.h"
```

Třídy

- class IceAge::Zbran

Prostory jmen

- IceAge

7.144 Zbran.h

```
00001
00004 #ifndef ZBRAN_H
00005 #define ZBRAN_H
00006
00007 #include"Predmet.h"
00008
00009
00010 namespace IceAge {
00014     class Zbran : public IceAge::Predmet {
00015
00016     private:
00017         short m_poskozeni;
00019         void setPoskozeni(short novePoskozeni);
00022     public:
00023         short getPoskozeni();
00024         short getAtribut();
00025         Zbran(std::string novyNazev, short novePoskozeni);
00026         ~Zbran();
00027     };
00028 }
00029 #endif
```

7.145 Dokumentace souboru hlavicky/Zivot.h

```
#include "IZivot.h"
#include "MemoryArbiter.h"
```

Třídy

- class IceAge::Zivot

Prostory jmen

- IceAge

7.146 Zivot.h

```
00001
00004 #ifndef ZIVOT_H
00005 #define ZIVOT_H
00006
00007 #include "IZivot.h"
00008 #include "MemoryArbiter.h"
00009
00010 namespace IceAge {
00014     class Zivot : public IceAge::IZivot {
00015
00016     protected:
00017         short m_zivotAktualni;
00018         short m_zivotMaximalni;
00020         void setZivotAktualni(short novyZivotAktualni);
00021         void setZivotMaximalni(short novyZivotMaximalni);
00023     public:
00024         void changeZivotAktualni(short hodnotaZmeny);
00026         short getZivotAktualni();
00028         short getZivotMaximalni();
00030         Zivot(short novyZivot, short novyMaximalniZivot);
00031         virtual ~Zivot();
00033         virtual short utok() = 0;
00035         virtual short obrana() = 0;
00036     };
00037 }
00038 #endif
```

7.147 Dokumentace souboru main.cpp

```
#include <iostream>
#include "hlavicky/EngineIceAge.h"
```

Funkce

- int main ()

7.147.1 Detailní popis

Tato hra je o veverce Scrat a jeho Orisku. Byla napsana jako skolni projekt do predmetu ZOO na MENDELU Hra je inspirovana filmem Doba Ledova.

Autor: Michal Srejber

Pocet radku zdrojovych souboru:3713

Definice v souboru main.cpp.

7.147.2 Dokumentace funkcí

7.147.2.1 int main ( )

<Spusteni samotneho hrani

<Vymazani veskere zbyvajici pameti

Definice je uvedena na řádku 13 v souboru main.cpp.

7.148 main.cpp

```
00001
00010 #include <iostream>
00011 #include "hlavicky/EngineIceAge.h"
00012
00013 int main()
00014 {
00015     IceAge::EngineIceAge *dobaledova=
      IceAge::EngineIceAge::getInstance();
00016
00017     dobaledova->startHry();
00019     IceAge::MemoryArbiter::cycleDelete();
00020     return 0;
00021 }
```