Προγραμματιστικές Τεχνικές
1η Εργασία:
"The Good, the Bad and the Zombie"

Παπαχρήστου Μάριος
ΣΗΜΜΥ ΕΜΠ / ΑΜ: 03115101
papachristoumarios@gmail.com
el15101@mail.ntua.gr

1 Περιγραφή της Εργασίας

Στην παρούσα εργασία, καλούμαστε να υλοποιήσουμε ένα παιχνίδι το οποίο θα περιέχει τρία είδη χαρακτήρων (Character) τους Good, Bad, Zombie, όπου καθένας από αυτούς έχει διαφορετικές ιδιότητες. Καθένας από τους χαρακτήρες έχει 100 hit points οι οποίοι μειώνονται κατα 10 όταν δέχεται επίθεση από κάποιον χαρακτήρα.

Οι υποστάσεις του τύπου Good επιτίθενται σε υποστάσεις τύπου Bad και τύπου Zombie. Η επίθεση σε τύπο Zombie έχει 20% πιθανότητα να γιατρέψει το Zombie και να το μετατρέψει σε καλό (Good) ($P_{Good} = 0.4$) είτε κακό (Bad) ($P_{Bad} = 0.6$). Οι υποστάσεις τύπου Bad επιτίθενται σε υποστάσεις Good και Zombie μειώνοντας τα hit points τους. Τέλος, οι υποστάσεις τύπου Zombie επιτίθεται σε Good και Bad με πιθανότητα 10% να τα μετατρέψουν σε Zombie.

Οι παίκτες τοποθετούνται σε μια κυκλική λίστα και ο καθένας κάνει επίθεση τον επόμενό του, συμπεριλαμβανομένων των ακραίων θέσεων. Το παιχνίδι τελειώνει είτε στους 100 γύρους είτε όταν επιζήσει μόνο μια από τις τρεις ομάδες σε λιγότερο από 100 γύρους.

Ο πηγαίος κώδικας του παιχνιδιού βρίσκεται στα αρχεία `src/characters.cpp` (source file) και `src/characters.h` (header file) στο φάκελο src. Επίσης παρέχεται και ένα Makefile[1] για την ευκολότερη παραγωγή εκτελέσιμων αρχείων σε συστήματα GNU / Linux. Με την εκτέλεση της εντολής `make` παράγονται τα εκτελέσιμα. Περισσότερα στοιχεία όσον αφορά την επιμέρους τεκμηρίωση των κλάσεων βρίσκονται στο αρχείο `characters.h`. Για τη διαχείριση της εργασίας χρησιμοποιήθηκε το Git. Το πρότυπο μορφοποίησης του κώδικα (ονοματοδοσία) που ακολουθήθηκε είναι το PEP 8[2].

2 Σχεδιασμός, Περιγραφή και Ιεραρχία Κλάσεων και Μεθόδων

2.1 Κλάση Character

Η βασική κλάση για τους χαρακτήρες του παιχνιδιού. Περιέχει ένα πεδίο με τους hit points του χαρακτήρα

1 https://www.gnu.org/software/make/
2 https://www.python.org/dev/peps/pep-0008/

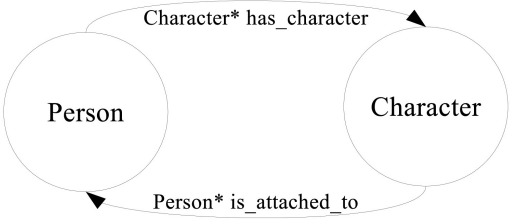
και ένα πεδίο-ταυτότητα (character ID) που λαμβάνει τιμές από τον απαριθμητό τύπο[3] (enum type) `{GOOD, BAD, ZOMBIE}`. Η κλάση αυτή παρέχει κάποιες θεμελιώδεις λειτουργίες και παίζει το ρόλο διεπαφής (interface) για την υλοποίηση των διαφόρων στοιχείων του παιχνιδιού. Για το λόγο αυτό οι περισσότερες μέθοδοι της κλάσης έχουν δηλωθεί εικονικές (virtual methods). Η κλάση διαθέτει επίσης όλους τους απαραίτητους προσπελαστές και τροποποιητές (accessors και mutators). Ο δομητής της κλάσης έχει οριστεί `protected`

2.2 Κλάσεις Good, Bad και Zombie

Κληρονομούν με δημόσια κληρονομικότητα (public inheritance) από την κλάση `Character` υλοποιώντας τη συγκεκριμένη διεπαφή. Ο κάθε τύπος χαρακτήρα έχει υλοποιημένη τη δική του μέθοδο `Attack(Character&)` που έχει δηλωθεί ως virtual στην μητρική κλάση `Character`. Επίσης η κάθε κλάση έχει δηλωμένους και τελεστές μετατροπής (cast operators) σε άλλους τύπους. Για παράδειγμα, στην κλάση `Zombie` είναι δηλωμένος ο `operator Bad()` ο οποίος μετατρέπει (cast) ένα `Zombie` σε `Bad`. Οι δομητές των συγκεκριμένων κλάσεων καλούν τον δομητή της Character με ένα `PlayerType` σαν όρισμα.

2.3 Κλάση Person

Η κλάση Person είναι μια κλάση-συσκευαστής (wrapper class) η οποία ορίζει μια αμφιμονοσήμαντη (bijection) σχέση με την κλάση `Character` μέσω του πεδίου `has_character` που είναι δείκτης σε `Character` (αντίστοιχο πεδίο `is_attached_to` της κλάσης `Character` το οποίο είναι δείκτης σε `Person`). Η κλάση αυτή δεν επιτελεί κάποιο λειτουργικό σκοπό παρά μόνο καθιστά ευκολότερη την διαχείριση των χαρακτήρων όταν αυτοι τοποθετηθούν σε κάποια λίστα. Η κλάση `Person` περιέχει επίσης τα hit points του συσκευασμένου χαρακτήρα ως έναν δείκτη σε ακέραιο. Τέλος, σημειώνεται ότι οι περισσότερες μέθοδοι που επιτελεί η κλάση `Character` έχουν συσκευαστεί μέσα στην `Person`.



3 Ορίζεται ως PlayerType

2.4 Κλάση World

Η κλάση `World` είναι η κλάση μέσα στην οποία υλοποιείται το παιχνίδι. Αρχικά, με τη βοήθεια της std::list έχουμε ορίσει έναν τύπο `PlayerList` ως `std::list<Person*>` ο οποίος αποτελεί μια συνδεδεμένη λίστα η οποία περιέχει δείκτες σε αντικείμενα τύπου `Person`. Επίσης η κλάση περιέχει τη μέθοδο `play_round()` η οποία εκτελεί ένα γύρο του παιχνιδιού και την κλάση `play_world()` η οποία εκτελεί ένα ολόκληρο παιχνίδι. Ο δομητής της κλάσης γεννά N χαρακτήρες και τους τοποθετεί μέσα στη λίστα `players` (ιδιωτική μεταβλητή τύπου `PlayerList`), ισοπίθανα όσον αφορά το είδος τους κάνοντας χρήση της `std::uniform_int_distribution` ως γεννήτρια (ψευδο)-τυχαίων αριθμών η οποία δίνει πιθανότητες στο διάστημα *[a,b]* σύμφωνα με τη συνάρτηση πυκνότητας πιθανότητας για τα ισοπίθανα ενδεχόμενα:

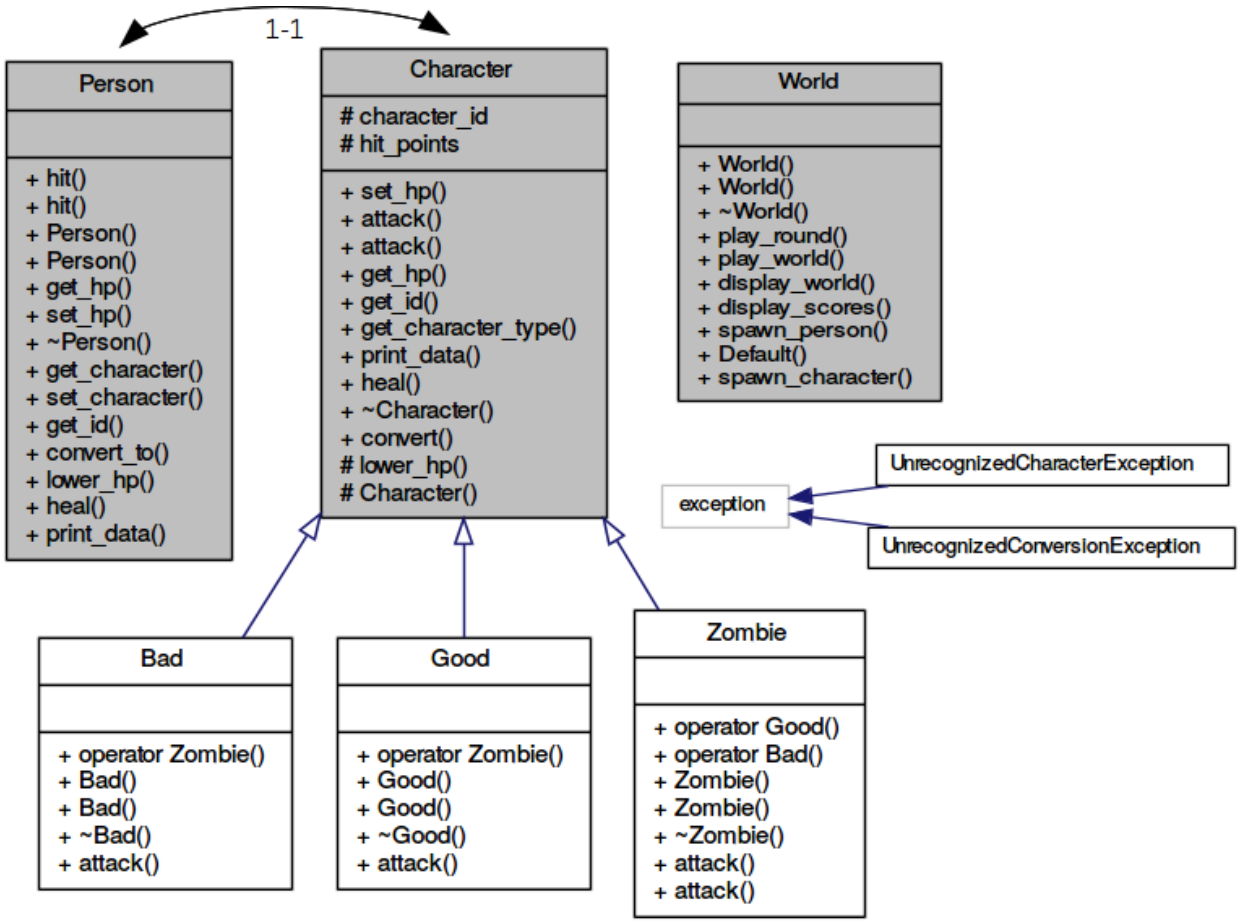
$$P(x=i,[a,b])=\frac{1}{b-a+1},\ i\in[a,b]$$

Τέλος, αναλυτικότερη εποπτεία του σχεδιασμού του προγράμματος μπορεί να έχει κάποιος με τα διαγράμματα **UML** που παρατίθενται στην επόμενη σελίδα.

2.5 Λοιπές Μέθοδοι

`Character* convert_to(Character*, PlayerType)`	Φίλια συνάρτηση (friend function) στην κλάση `Chararcter` η οποία επιτρέπει την μετατροπή ενός `Character` σε κάποιο άλλο τύπο. Η ουσία της μεθόδου έγκειται στη δημιουργία ενός νέου δείκτη σε `Character` και τη διαγραφή του υπάρχοντος αφού πρώτα "μεταβιβαστούν" τα hit points στο νέο `Character`. Τέλος επιστρέφεται ο δείκτης στο νέο `Character.`
`Person* convert_to(Person*, PlayerType)`	Η αντίστοιχη ορίζεται για μετατροπή `Person*` και είναι φίλια στην `Person`
`Character& Character::convert(PlayerType)`	Προστατευμένη (protected) μέθοδος της κλάσης `Character` η οποία επιτρέπει την μετατροπή του ίδιου του αντικειμένου (τροποποίηση του `*this`) καλώντας την `Convert` έτσι το αποτέλεσμα της `Convert` εναποτίθεται στο `*this` αλλάζοντας "αυτοπαθώς" τον τύπο του στιγμιοτύπου.

2.6 Ιεραρχικό Διαγράμμα (Hierarchical Diagram) κατά UML[4]



3 Χρήση

3.1 Δημιουργία Χαρακτήρων

Με χρήση της κλάσης-συσκευαστή Person:

```
Person *p = new Person(GOOD);
Person *q = new Person(ZOMBIE);
p->hit(q); //p attacks q
q->print_data();
```

Με χρήση της κλάσης Character:

```
Character *g = new Good();
Character *z = new Zombie();
g->attack(z); //g attacks z
z->print_data();
```

3.2 Δημιουργία Κόσμου

```
World *world = new World(10);
world->play_round(); //plays one round
world->display_world();
world->play_world(); //plays a whole game
delete world;
```

4 Τα διαγράμματα ιεραρχίας κατά UML έγιναν με τη χρήση του Doxygen