Programación 4

EXAMEN JULIO 2014
SOLUCIÓN

Problema 1

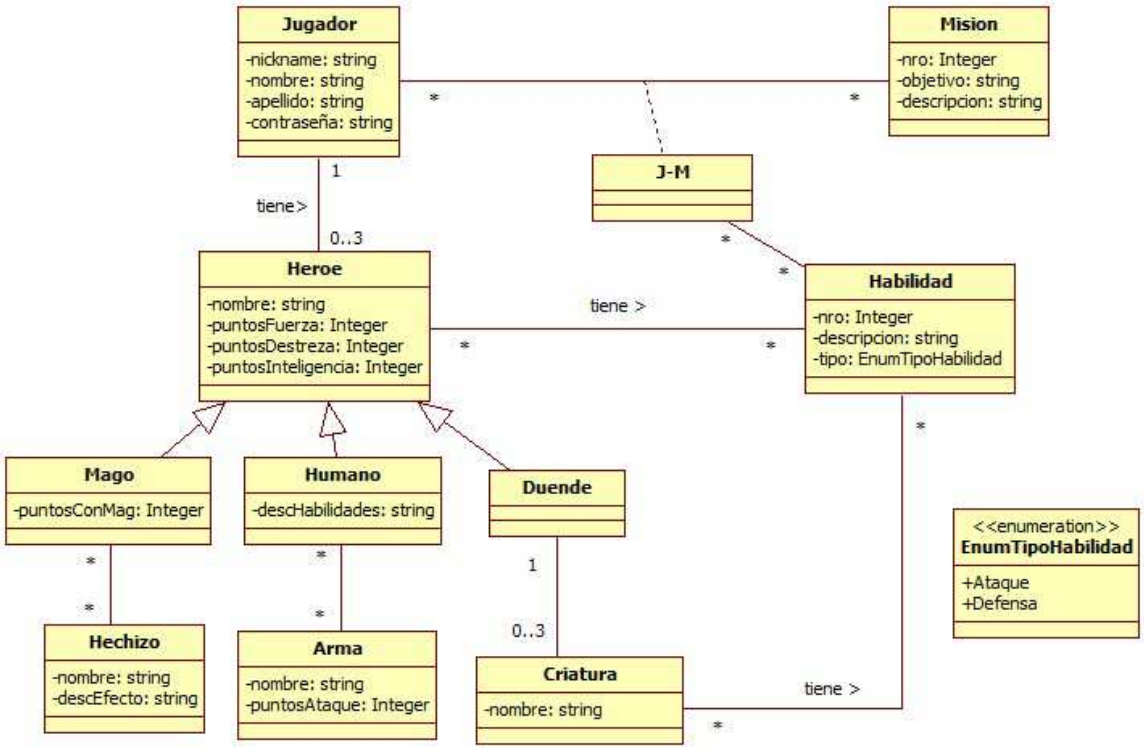
Parte A:

a) Un contrato de Software especifica el comportamiento o efecto de una operación (Diapositiva 39 del Teórico 06).

b) Una clase es un descriptor de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y comportamiento (Diapositiva 8 del Teórico 03).

Parte B:

a)



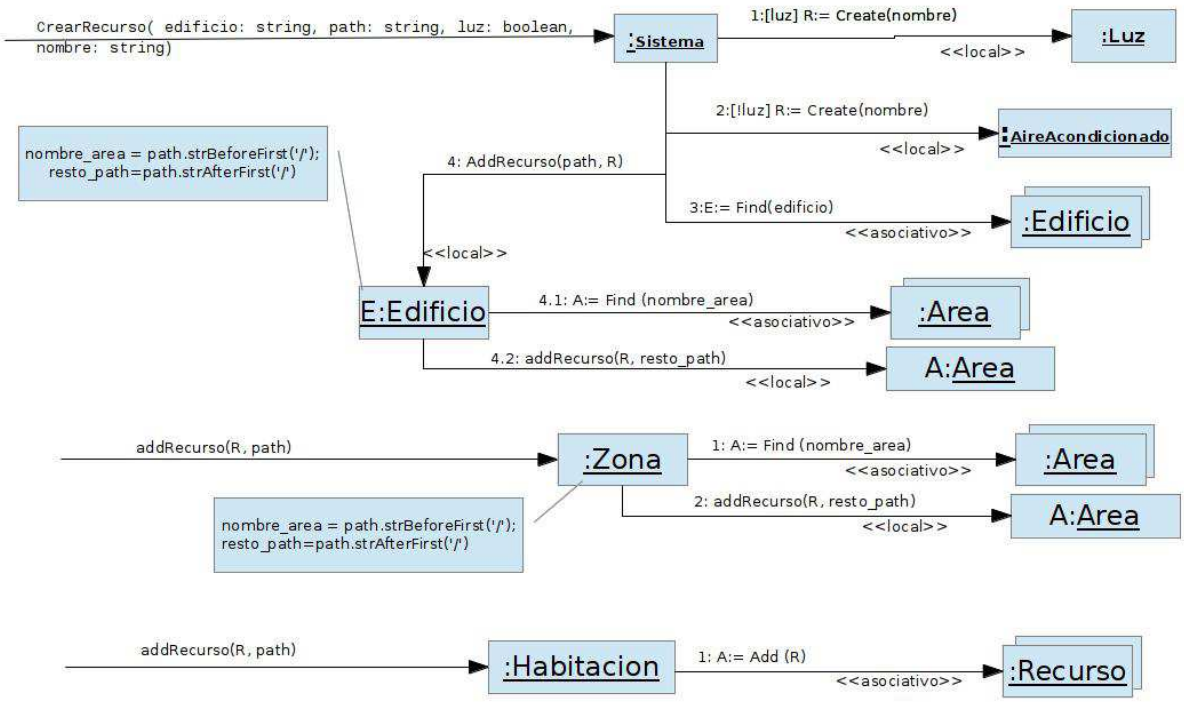
b)

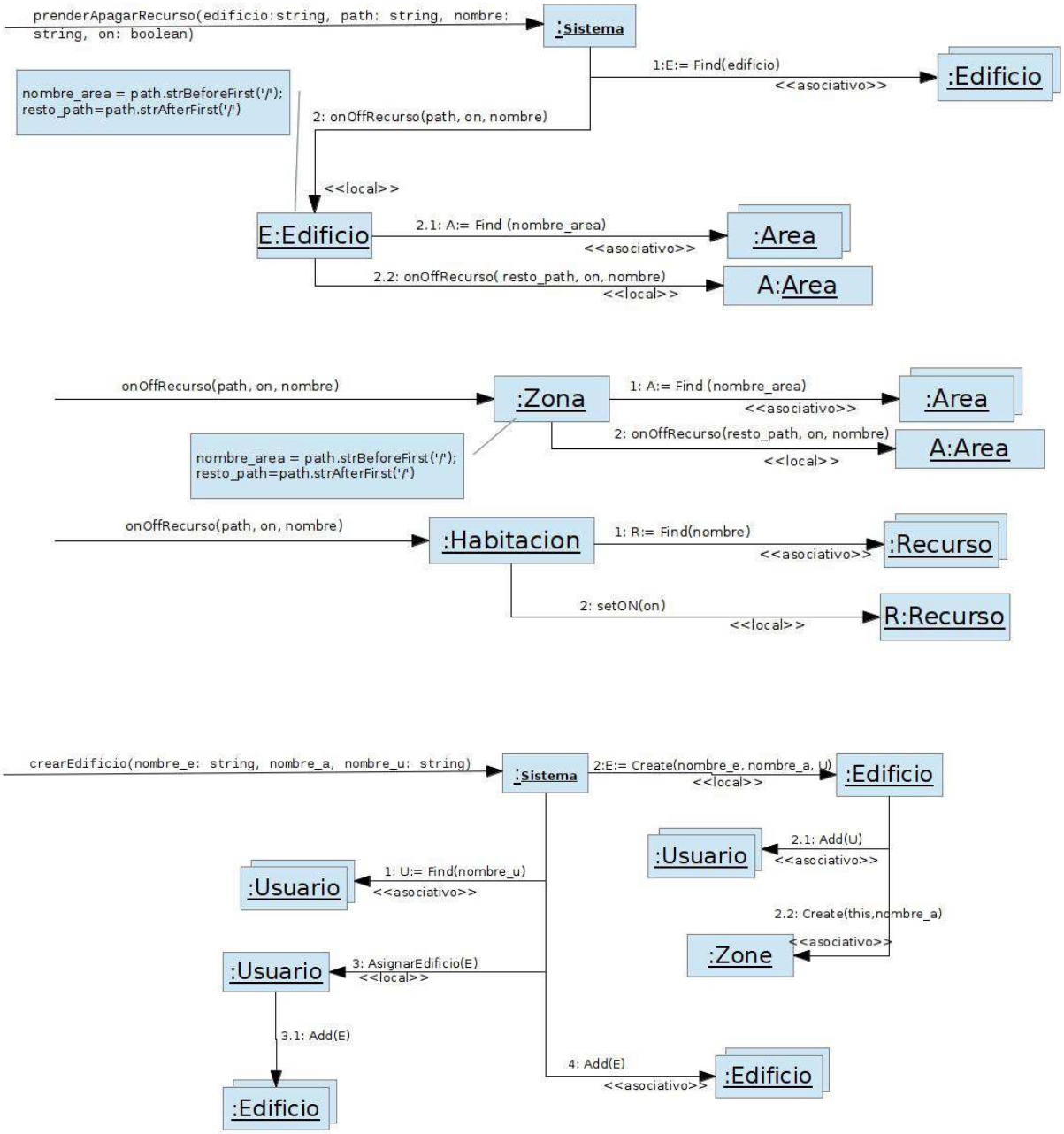
1- Un jugador se identifica por su nickname.
2- Un héroe se identifica por su nombre.
3- Una habilidad se identifica por su número.
4- Un hechizo se identifica por su nombre.
5- Una criatura se identifica por su nombre.
6- Un arma se identifica por su nombre.
7- El nombre de una criatura está compuesto por el nombre de la instancia de Duende que la invocó más caracteres adicionales.

8- Dado un Duende que invoca una Criatura, que posee una habilidad, entonces el duende no posee dicha habilidad.
9- Dada una instancia de tipo asociativo J-M de un Jugador y de una Misión, asociado a una Habilidad de un Héroe. Ese Héroe está asociado al Jugador antes mencionado.

Problema 2

Parte A:
a)




prenderApagarRecurso(edificio:string, path: string, nombre: string, on: boolean)
:Sistema
1:E:= Find(edificio)
<<asociativo>>
:Edificio
nombre_area = path.strBeforeFirst('/');
resto_path=path.strAfterFirst('/')
2: onOffRecurso(path, on, nombre)
<<local>>
E:Edificio
2.1: A:= Find (nombre_area)
<<asociativo>>
:Area
2.2: onOffRecurso(resto_path, on, nombre)
<<local>>
A:Area
onOffRecurso(path, on, nombre)
:Zona
1: A:= Find (nombre_area)
<<asociativo>>
:Area
2: onOffRecurso(resto_path, on, nombre)
<<local>>
A:Area
nombre_area = path.strBeforeFirst('/');
resto_path=path.strAfterFirst('/')
onOffRecurso(path, on, nombre)
:Habitacion
1: R:= Find(nombre)
<<asociativo>>
:Recurso
2: setON(on)
<<local>>
R:Recurso

crearEdificio(nombre_e: string, nombre_a, nombre_u: string)
:Sistema
2:E:= Create(nombre_e, nombre_a, U)
<<local>>
:Edificio
:Usuario
2.1: Add(U)
<<asociativo>>
:Usuario
1: U:= Find(nombre_u)
<<asociativo>>
2.2: Create(this,nombre_a)
<<asociativo>>
:Zone
:Usuario
3: AsignarEdificio(E)
<<local>>
3.1: Add(E)
4: Add(E)
<<asociativo>>
:Edificio
:Edificio

Parte B:

a)

- Ver teórico. Presentación: 15 - disenio – patrones.pdf
- Otro ejemplo, strategy, proxy, template method.

b) El patrón usaso es composite.

- Area es el Componente
- Zona es el Compuesto
- Habitacion es la Hoja

Problema 3

Parte A:

```
void main (){
     Objeto* e = new Objeto();
     return;
}
```

Parte B:

```
// Timlinee.hh

class Timeline : public List{
private:
     int max_size;
     int current_size;
     Node* first;
public:
     Timeline(int);
     virtual ~Timeline();
     void add(Objet*);
     void remove(Objet*);
     Iterator* newsIterator(int);
     bool isEmpty();
}

// Timeline.cc

Timeline::Timeline(int ms){
     max_size = ms;
     current_size = 0;
}

Timeline::~Timeline(){
     Node* it, borrar;
     it = first;
     while(it!=NULL){
          borrar = it;
          it = it->getNext();
```

```
            delete borrar;
      }
}

void Timeline::add(Objet* o){
      Node* aux = first;
      Node* prev = new Node(o, aux);
      first = prev;
      current_size++;
      if (current_size > max_size){
            current_size--;
            aux = first;
            while(aux!=NULL){
                  if(aux->getNext()->getItem()->getTimestamp() <
                  prev->getNext()->getItem()->getTimestamp()){
                        prev = aux;
                  }
                  aux = aux->getNext();
            }
            if(first->getItem()->getTimestamp() < prev-getNext()-
                              >getItem()->getTimestamp()){
                  first = first->getNext();
                  delete first;
                  return;
            }
            aux = prev->getNext();
            prev->setNext(prev->getNext());
            delete aux;
      }
}

void Timeline::remove(Object* o){
      Node* it, next;
      it = first;
      if(it->getItem()==0){
            first = it->getNext();
            delete it;
            return;
      }
      while((it != NULL)){
            next = it->getNext();
            if((next != NULL) && (next->getItem() == o)){
                  it->setNext(next->getNext());
                  delete next;
                  return;
            }
            else
                  it = it->getNext();
      }
}

Iterator* Timeline::newsIterator(int from){
      return new NewsIterator(from);
}
```

```
bool Timeline::isEmpty(){
     return (first == NULL);
}

// Node.hh

class Node{
private:
     Object* item;
     Node* next;

public:
     Node(Node*, Object*);
     virtual ~Node();
     bool hasNext();
     Node* getNext();
     void setNext(Node*);
     Object* getItem();
}

// Node.cc
Node::Node(Object* o, Node* n){
     item = o;
     next = n;
}

bool Node::hasNext(){
     return (next == NULL);
}

Node* Node::getNext(){
     return next;
}

void Node::setNode(Node* n){
     next = n;
}

Object* Node::getItem(){
     return item;
}

Node::~Node(){
}

// Iterator.hh

class Iterator{
public:
     virtual bool hasNext() = 0;
     virtual Object* next() = 0;
     virtual ~Iterator(){};
}
```

```
// NewsIterator.hh

class NewsIterator : public Iterator{
private:
      int from;
      Node* current;

public:
      NewsIterator(int, Node*);
      ~NewsIterator();
      bool hasNext();
      Object* next();
}

// NewsIterator.cc

NewsIterator::NewsIterator(int f, Node* c){
      from = f;
      current = c;
}

NewsIterator::~NewsIterator(){
}

bool NewsIterator::hasNext(){
      if (current == NULL)
            return false;
      Node * tmp = current;

      while (tmp!=NULL){
            if (tmp->getItem()->getTimestamp()>from)
                  return true;
            tmp = tmp->getNext();
      }
      return false;
}

Object* NewsIterator::next(){
      Node * tmp = current->getNext();
      while (tmp!=NULL){
            if (tmp->getItem()->getTimestamp() > from){
                  current = tmp;
                  ret = tmp->getItem();
                  break;
            }
            else
                  tmp = tmp->getNext();
      }
      return current->getItem();
}
```