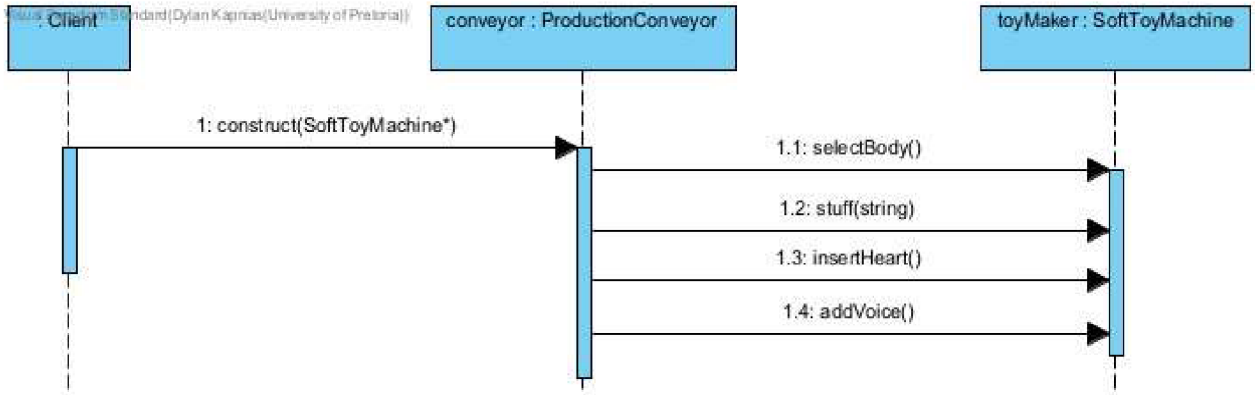
Question 1

i.) Memento.
ii.) Decorator.
iii.) Composite.
iv.) Observer.
v.) Iterator.

Question 2

a.) It is a dependency relationship as ProductionConveyor depends on SoftToyMachine to exist (i.e. be constructed).

b.)



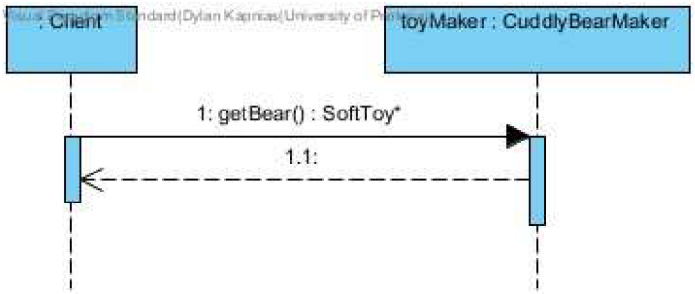
c.) i.) It represents the period of time that the object instantiated by the CuddlyBearMaker is in memory for.

```
SoftToy* toy = new SoftToy();
```

ii.)

```
for (int i = 0; i < 6; i++) {
        toy->show();
}
delete conveyor;
```

d.)



Question 3

a.) i.) watch my_var
ii.) The debugging will pause and it will display the variables old and new values, newline separated. When the command is used it will display “Hardware watchpoint x: my_var” where x depends on how many watchpoints have already been set.

b.) It is saying the program executing with process_id of 20688 has written 4 bytes to memory that was outside of the allocated block.

c.)
```
try {
        multiply(5,0);
        FAIL();
} catch (std::exception &err) {
        break;
}
```

Question 4

({Composite and Decorator and Template}, {Component and Component and AbstractClass}, Graphic)

({Composite and Decorator and Template}, {Leaf and ConcreteComponent and ConcreteClass}, Ellipse)

(Composite, Composite, CompositeGraphic)

(Decorator, Decorator, GraphicDecorator)

(Decorator, ConcreteDecorator, {Label and Box})

Question 5

a.)
```
#ifndef BOX_H
#define BOX_H

#include "GraphicDecorator.h"
#include <string>

class Box : public GraphicDecorator
{
   private:
      std::string _box;

   public:
      Box(Graphic*, std::string);
      virtual void print() override;
      Box();
};

#endif
```

b.) i.)

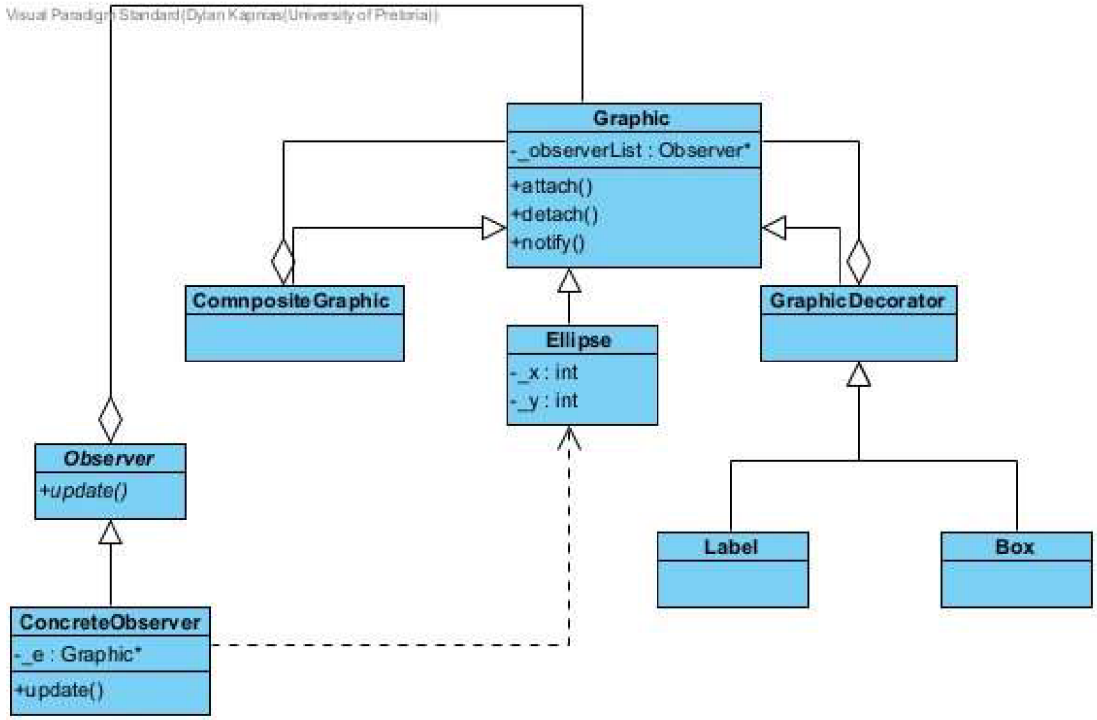
ii.)
```
Graphic* g2 = new CompositeGraphic();
    Graphic* e3 = new Ellipse(1, 33, 7, 12);

    e3 = new Box(e3, "Box");
    g2−>addGraphic(e3);
    g2 = new Label(g2, "Decorated");

    g->addGraphic(g1);
    g->addGraphic(g2);
```

c.) GraphicDecorator::~GraphicDecorator() {
    delete _component;
}

CompositeGraphic::~CompositeGraphic() {
    for (std::list <Graphic*>::iterator it = _l.begin(); it != _l.end(); ++it)
        _l.erase(it);
}

Question 6

a.) It needs to be defined as a virtual function in Graphic and implemented in Ellipse due to these being part of the template method.
b.) Subject
c.) Ellipse
d.) Graphic
e.)



Question 7

a.) A variable to point to the next Graphic* in the list, as well as one to point to the previous one in the list, and a function to create the iterator.

f.) *for* (GraphicIterator i = g2->createIterator(); !(i == g->last()); ++i)
        std::cout << *i->elementType() << std::endl;