Poznan University of Technology
Objects and classes

Jakub Piotr Hamerliński

Objects and classes

Agenda

01. Methods
02. Evilness of ~~getters~~ and ~~setters~~
03. Encapsulation
04. Task

> *"Me mechanic not speak English. But he know what me mean when me say "car no go", and we best friends. So me think: why waste time, say lot word when few word do trick?"* Kevin Malone

Methods

Methods

Introduction

The message is a request to an object to invoke one of its methods. The message contains:

01. The name of the method,
02. The arguments of the method.

A method is associated with a class. An object invokes one of its class methods as a reaction to the message

Methods

Examples

```
int Length(...);
std::string ParsedText(...);
void Save();
void Print();
```

Methods

More advance example - ball

```
class Ball {
 private:
  std::string color;
 public:
  Ball Paint(std::string newColor) {
    this->color = std::move(newColor);
    return *this;
  }
  std::string CurrentColor() {
    return this->color;
  };
};
```

Methods

More advance example - dog

```
class Dog {
 private:
  std::string name;
  std::optional<std::vector<Ball>> dogsBalls;
 public:
  void NameDog(std::string dogsNewName) {
    this->name = std::move(dogsNewName);
  }
  void GiveBall(const Ball &ball) {
    this->dogsBalls->push_back(ball);
  }
  Ball TakeBall() {
    Ball takenFromDog = this->dogsBalls->back();
    this->dogsBalls->pop_back();
    return takenFromDog;
  }
};
```

Methods

More advance example - usage

```
int main() {
  Dog goodBoy;
  goodBoy.NameDog("Piernik");
  Ball ball;
  Ball newBall = ball.Paint("Pink");
  goodBoy.GiveBall(newBall);
  Ball wetBall = goodBoy.TakeBall();
  std::cout << wetBall.CurrentColor();
  return 0;
}
```

Encapsulation

encapsulation (noun, BrE /ɪnˌkæpsjuˈleɪʃn/):
the act of expressing the most important parts of something in a few words,
a small space or a single object

Oxford Dictionary

Encapsulation

Examples

```
class Book {
 private:
  std::string isbn;
  std::string author;
  std::string title;
  int pages;
};
```

```
class Article {
 private:
  Doi doi;
  Author* authors;
  Title title;
  int pages;
};
```

Encapsulation

More technical explanation

The fundamental idea behind object-oriented computing is encapsulation, which gives objects their solidity, coherence, reliability, etc. But what does encapsulation actually mean? Does it only defend against external access to confidential attributes? It is much much more. All layers and forms of naked data are eliminated through encapsulation.

Evilness of ~~getters~~ and ~~setters~~

Evilness of ~~getters~~ and ~~setters~~

Why they are evil?

“It is convenient to have these ~~getters~~, you may say. We are all used to them. If we want to convert it into JSON, they will be very helpful. (...) There are many examples, where ~~getters~~ are being actively used. This is not because they are so effective. This is because we’re so procedural in our way of thinking. We don’t trust our objects. We only trust the data they store. ”
Yegor Bugayenko

Evilness of ~~getters~~ and ~~setters~~

Why they are evil?

Violated Encapsulation Principle
Exposed Implementation Details
Telling Instead Of Asking

Evilness of ~~getters~~ and ~~setters~~

Violated Encapsulation Principle

They break encapsulation:

```
1   // wrong example below
2   class Temperature {
3    private:
4     int t;
5    public:
6     void SetTemp(int newTemp){
7       this->t = newTemp;
8     }
9     int GetTemp(){
10      return this->t;
11    }
12  };
```

Evilness of ~~getters~~ and ~~setters~~

Violated Encapsulation Principle

Better do it like this:

```
// good example below
class Temperature {
 private:
  int t = 12;
 public:
  std::string FahrenheitValue() {
    return std::to_string(t) + " °F";
  }
};
```

Evilness of ~~getters~~ and ~~setters~~

Exposed Implementation Details

Usage of getters, setters leads to data structures/bags, not to classes:

```
1  struct Cash {
2    int dollars;
3  }
4  printf("Cash value is %d", Cash.dollars);
```

Evilness of ~~getters~~ and ~~setters~~

Telling Instead Of Asking

Use printers instead of getters:

```
class Book {
 private:
  std::string isbn = "0735619654";
  std::string title = "Object Thinking";
 public:
  std::string JSON() {
    return "{\"isbn\":\"" + this->isbn + "\", \"title\":\"" + this->title + "\"}";
  }
};
```

Task

Methods

Task

Task

01. Write a program in C++, which will demonstrate communication between different objects

Thank you

Feel free to reach me via LinkedIn

Fin