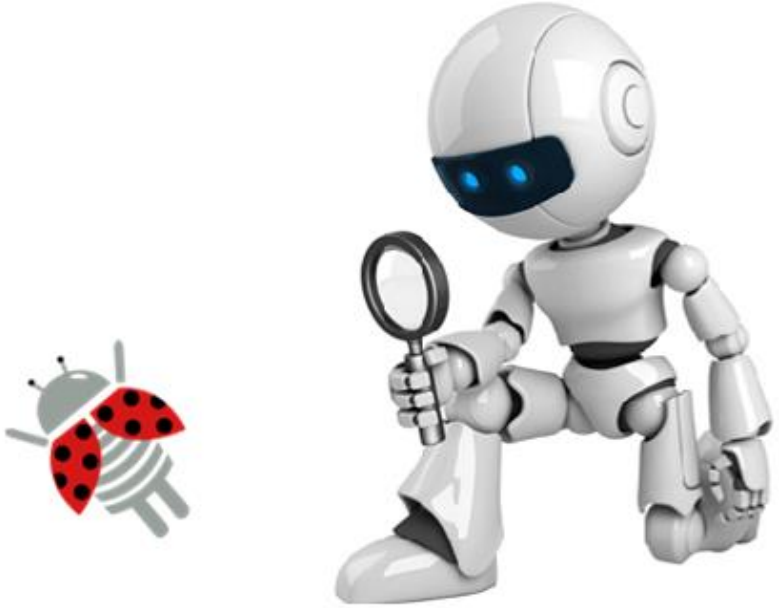
QATestify.com

Test automation course



Lesson 1

Table of contents

Installing software

1. Navigate to https://www.java.com/en/download/manual.jsp and install Java (using your platform and operating system)
2. Navigate to https://www.eclipse.org/downloads/ and install Eclipse
3. Navigate to http://www.xmind.net/download/win/ and install xMind
4. Navigate to http://mozilla-firefox.en.softonic.com/download and install latest Firefox
5. In Firefox install https://addons.mozilla.org/ru/firefox/addon/firepath/ FirePath extension

Installing software

6. Navigate to http://getfirebug.com/ and install Firebug
7. Install Jing (https://www.techsmith.com/jing.html) for screenshots and short videos, oCam (http://ohsoft.net/en/product_ocam.php) or any other tool for long videos
8. Set up your Git repository (https://www.youtube.com/watch?v=73I5dRucCds), send its name, user and password to lecturer
9. Install Team Viewer

Insight to test automation

Automation is the use of tools and strategies that reduce human involvement or interaction in unskilled, repetitive or redundant tasks

Automation is the use of machines or technologies to optimize productivity in the production of goods and delivery of services

Automation is used to increase productivity, and/or quality beyond that possible with current human labour levels so as to realize economies of scale, and/or realize predictable quality levels

To understand **test automation** in a nutshell, start with great video from Guru99.com https://www.youtube.com/watch?v=RbSlW8jZFe8

When test automation should be used

- ✓ Expanding regression suites
- ✓ Rapid release-cycles
- ✓ Test complexity
- ✓ Repetitive tests that run for multiple builds
- ✓ Tests that are highly subject to human error
- ✓ Tests that require multiple data sets
- ✓ Frequently-used functionality that introduces high risk conditions
- ✓ Tests that are impossible to perform manually
- ✓ Tests that run on several different hardware or software platforms and configurations
- ✓ Tests that take a lot of effort and time when doing manual testing

When test automation should not be used

- ✓ Unstable requirements
- ✓ Short Project
- ✓ Low ROI
- ✓ Technically impossible

Test automation classification

Semi – automatic testing expects user intervention at a certain point of test execution to add data, change settings, verify outcome and tune the process.

Test automation classification

Automatic testing expects no user intervention during test execution, however its preparation and maintenance requires more time and resources.

Types of testing to be automated

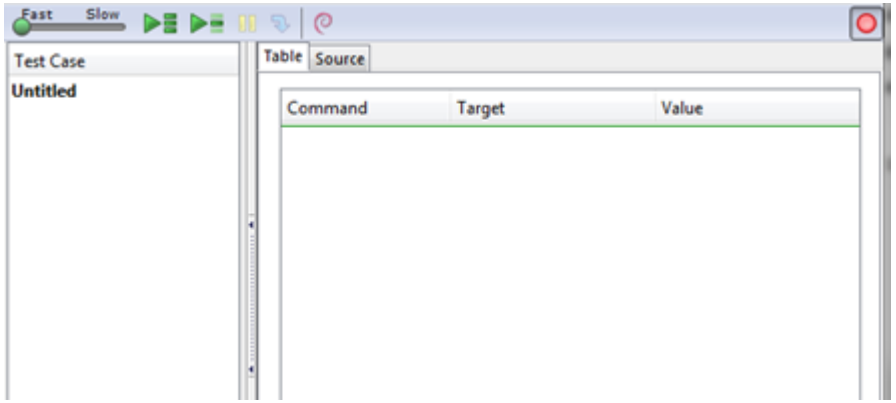
- ✓ Unit testing
- ✓ Sanity/Smoke testing
- ✓ Regression testing
- ✓ Functional testing
- ✓ Load/Stress testing
- ✓ GUI testing
- ✓ UAT testing
- ✓ API testing
- ✓ Web services testing

Approaches in test automation

Record / Playback

Record / Playback approach records your keystrokes and gestures while you are testing an application. The next time you run the test, you can play back your actions quickly and accurately.

Playback is very useful for reproducing bugs. Exact actions can be retraced to the point where the fault was discovered.

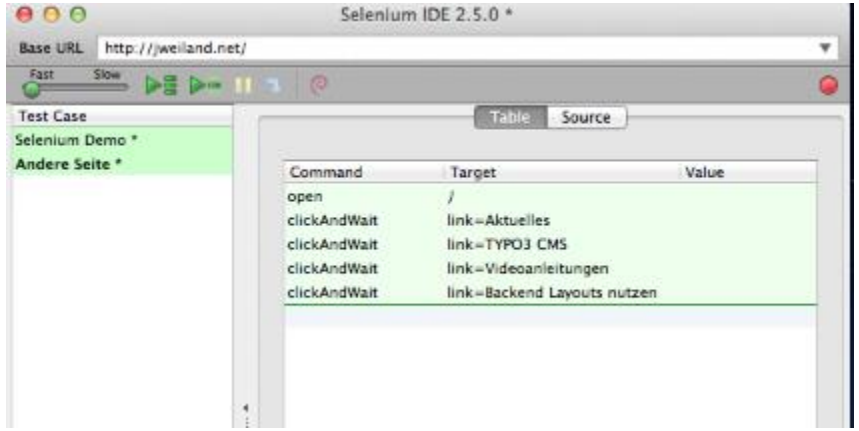


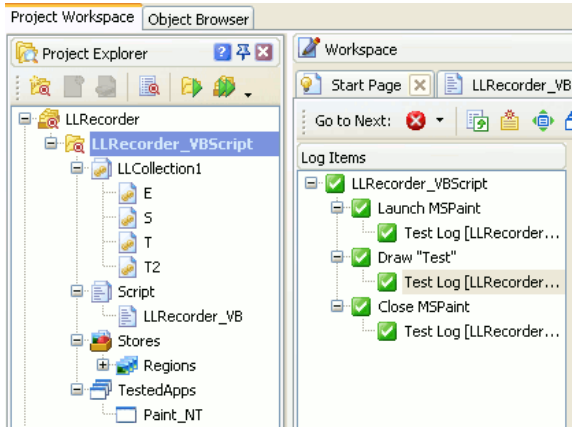
Approaches in test automation

Record / Playback enhanced

This is logical extension of Record / Playback approach with different data, spreading to multiple configurations.

It is possible to create shared steps to be used in different test cases, re-use steps by copying/pasting/modifying.

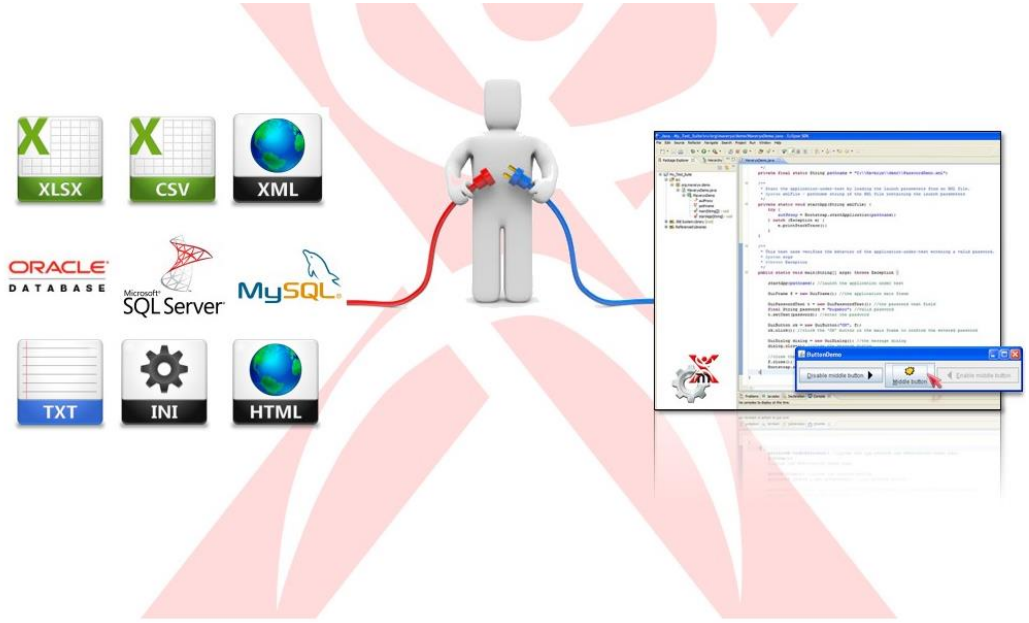




Approaches in test automation

Data-driven

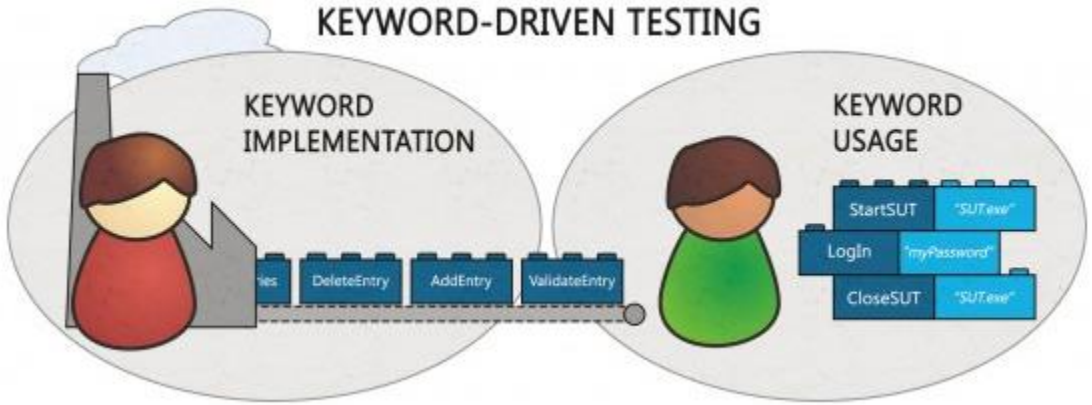
This is approach which expects storage of test data separately from scripts.



Approaches in test automation

Keyword-driven

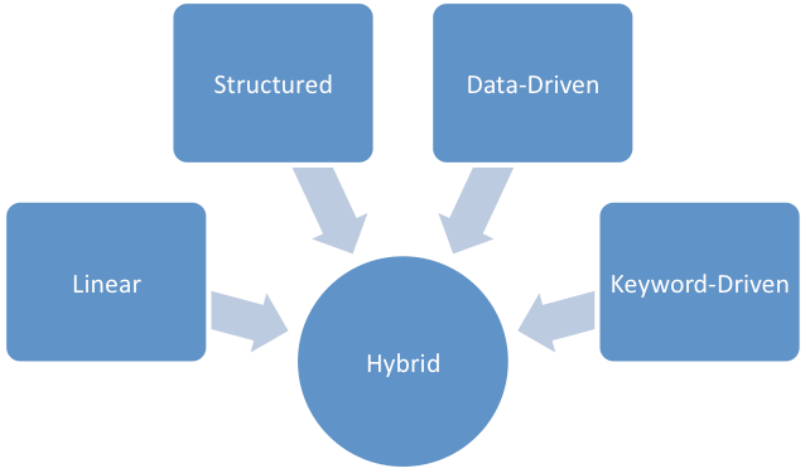
The basic idea behind the keyword-driven testing approach in test automation is to separate the automation from the test case design. Designed mostly for software testers with little programming background. Keywords are to be defined as sets of actions performing a “Login”, “Save” or “Enter data” operations.



Approaches in test automation

Hybrid

As a result of record/playback (linear), record/playback enhanced (structured), data-driven and keyword-driven combination, hybrid approach represents what is usually called “Test Harness” or “Test Automation framework”



Myths of test automation

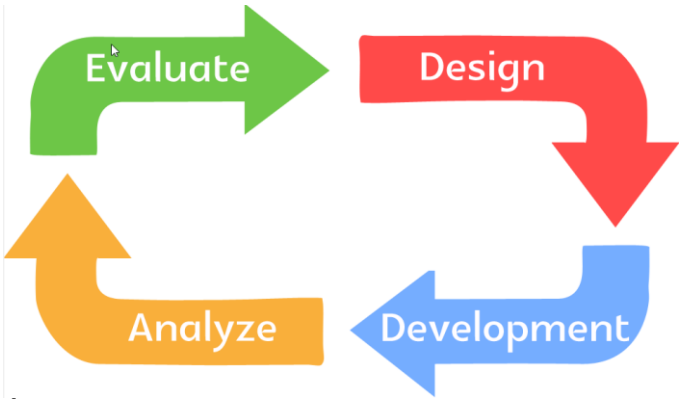
- ✓ No need to do automation, everything can be checked manually
- ✓ There is state of art in manual testing, test automation is just programming
- ✓ It is easy to do automation
- ✓ Customers are not interested to pay for automation
- ✓ It is possible to do automation only within project scope

- ✓ If everything can be automated - all manual testers will be fired from the project
- ✓ Salary of test automation engineers is pretty much the same considering higher effort

How to plan test automation (TA)

A. Initial planning and investigations

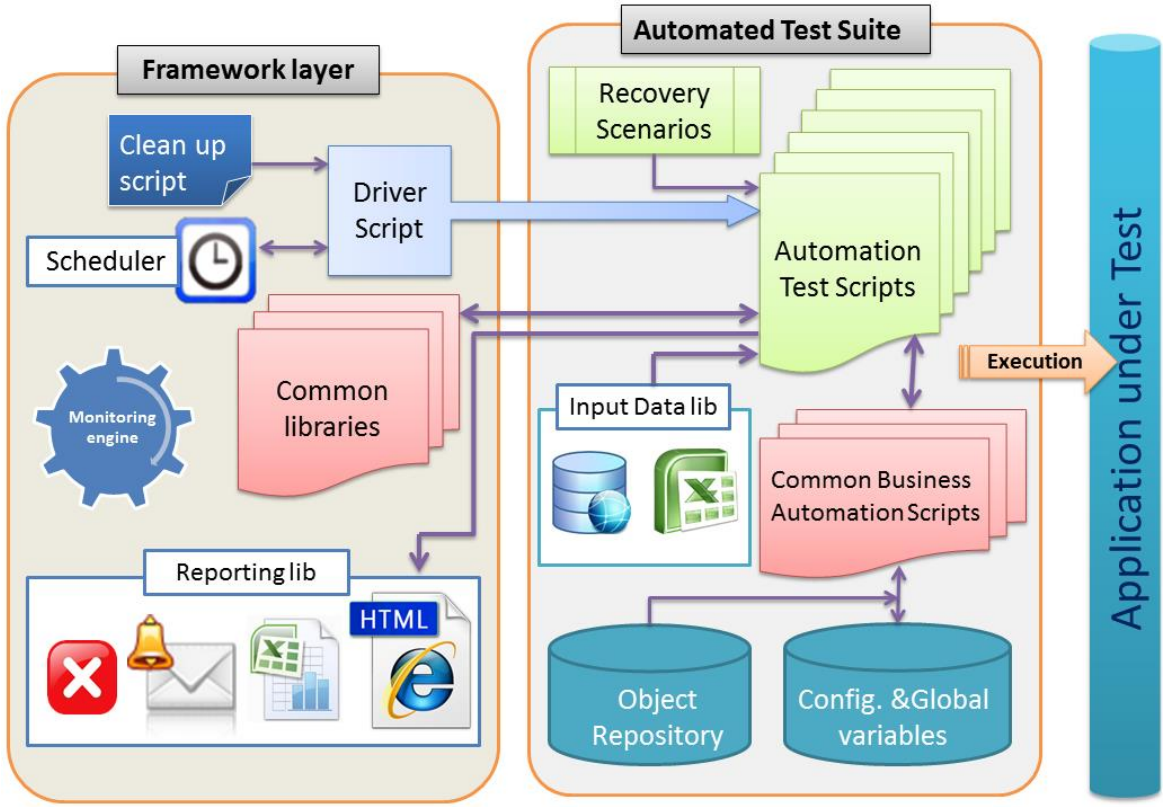
1. Resources - decide who will do TA
2. Tools - decide which tool and/or technology will be used
3. Time – estimate how much time needed
4. Money – what is the budget
5. Time – when TA should be started
6. Risks – what can be an issue during test automation



B. Start test automation activities

C. Calculate ROI (return on investment)

Test Automation framework



Object – oriented programming basics

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects“:

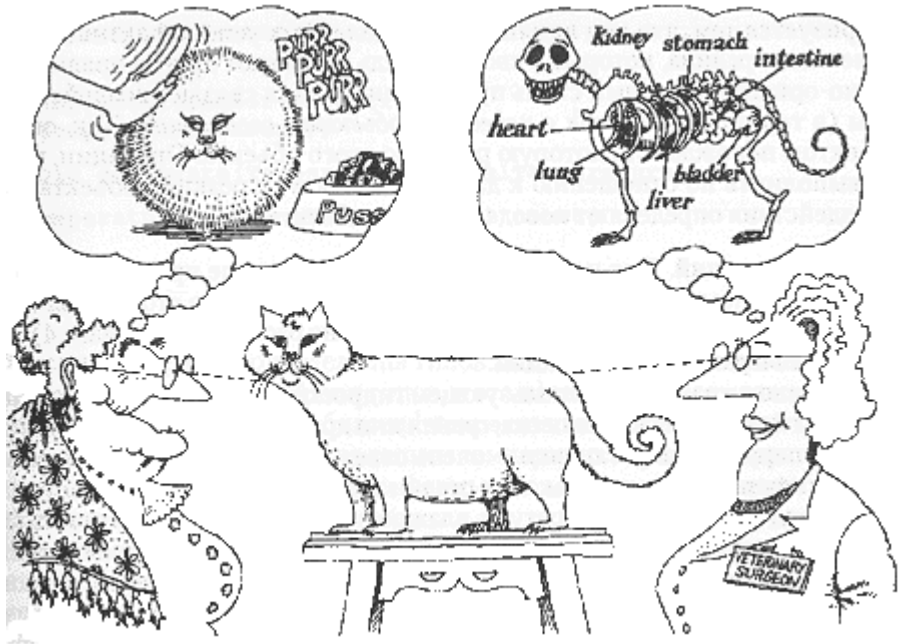
- ✓ Object is set of data structures which contains data in the form of fields, often known as attributes
- ✓ Object contains code, in the form of procedures, often known as methods

Object – oriented programming basics

- ✓ Key principle to understand - distinguishing feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated
- ✓ In object-oriented programming, computer programs are designed by making them out of objects that interact with one another
- ✓ Most popular languages are class-based, meaning that objects are instances of classes, which typically also determines their type

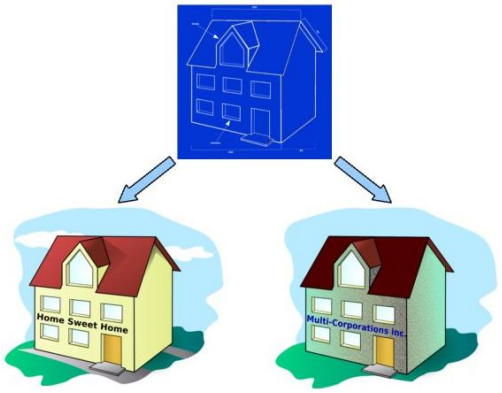
Object – oriented programming basics

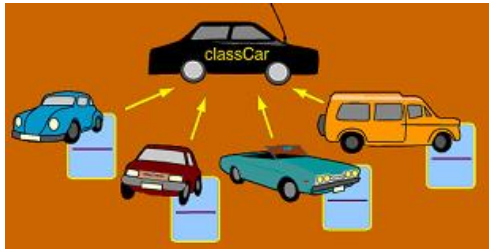
Difference between normal cat and OOP cat



Object – oriented programming basics

Class is an extensible program-code-template for creating objects, providing initial values for state (member variables) and implementations of behaviour (member functions, methods)





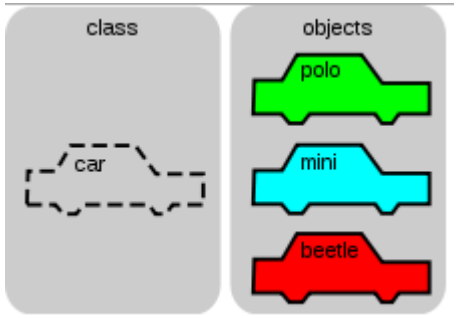
Object – oriented programming basics

Class should be treated as very general structure definition which later will be extended to certain level of details.

Class is general way to describe some instances. Class description requires existence of states and correspondent behaviours, which actually depend on these states. Also rules to interact with such instances have to be defined.

Classes examples: Animals, Humans, Buildings, Vehicles, Companies, Computing machines, Workers, Entertainments etc.

Obviously it sounds quite complex. Detailed examples will do explanation



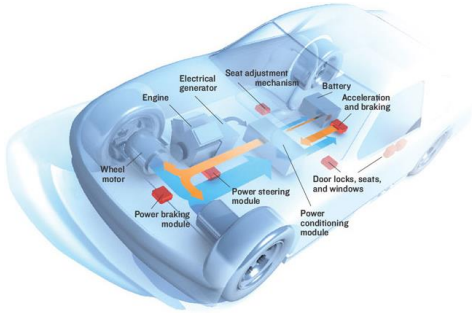
Object – oriented programming basics

Class “Car”

Attributes:

year of manufacture, type of car (sedan, hatchback, SUV, cabriole, off-road), number of doors (2,3,5,6,7), number of wheels (4,6,8), gear type, colour, vehicle volume, vehicle power, additional options, number of SRS airbags, fuel type, wheel size, tyres type, tyres size, trunk volume, speed

States:



moving forward, moving backward, moving on neutral, standing, speeding, breaking, turning, drifting, signalling, making smoke, not operating

Object – oriented programming basics

Class “Car”

Methods part 1: clean car, repair car, open door, open trunk, close door, close trunk, turn engine on, press accelerator pedal, press break pedal, turn lights on/off

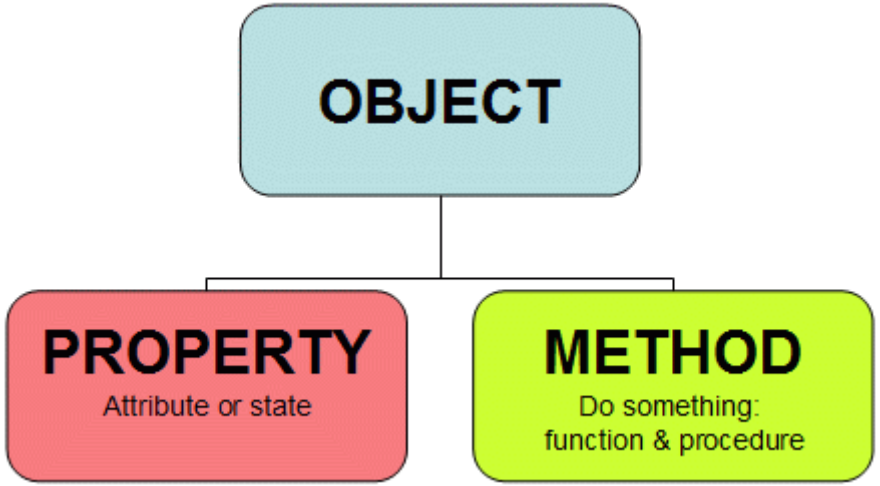
Methods part 2: push fuel to engine, enlarge pressure to put cleaning liquid on wind protection glass, switch lights automatically when dark, turn on rain cleaners automatically, turn on ABS, activate airbags, turn parking sensors on, allow wheels spin, turn “Low fuel” indicator

Rules: Methods part 1 are available for class “Driver” to be executed.
Methods part 2 are internal and available only inside “Car” class

Object – oriented programming basics

Object is separate class representative which has definite state and behaviour fully defined by the class

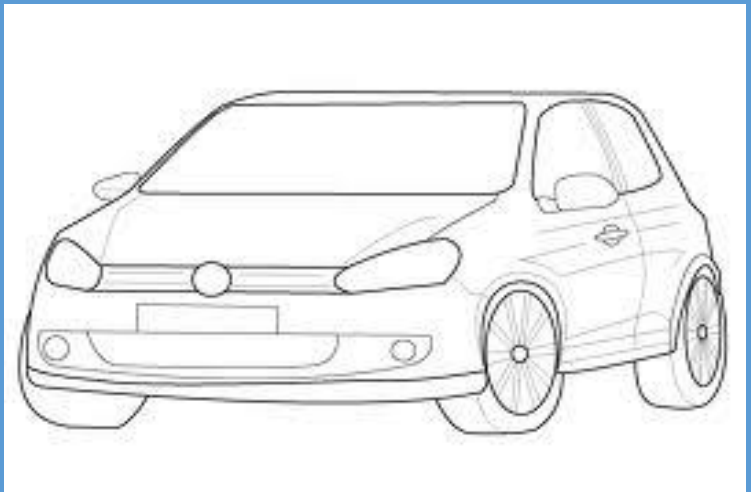
Object has properties (data) and methods (code)



Object – oriented programming basics

Difference between class and object

Class “Car” is some abstract concept created to serve as template from “world of ideas”



Object “Car” is quite definite car standing near your house and ready to be driven

Object – oriented programming basics

Abstraction is simplified object description. Main idea is that some object properties have to be defined while other have to be skipped.

Correct selection of abstractions sets treated as main task of object-oriented programming

```
public class Animals
        {
          int age;
          int weight;
          int height;
          String color;
          boolean isWool;
        }
```

```
public class Buildings
        {
          int yearOfConstruction;
          String buildingType;
          int floorQuantity;
          String color;
          boolean isElevator;
        }
```

Object – oriented programming basics

Encapsulation allows to hide class information, access to which should not be allowed by other classes and their objects.

Access modifiers are representing encapsulation nature. For the purposes of this course such should be known:

public modifier expects that declaration which it follows is available for everybody

private modifier expects that declaration which it follows is available only inside methods of current class

protected modifier expects that declaration which it follows is available only inside methods of current class and classes inherited from this class

Object – oriented programming basics

Abstraction is responsible for external object behaviour	**Encapsulation** is responsible for internal object structure and behaviour
BEFORE *public class Animals {* *public int age;* *public int weight;* *public int height;* *public String color;* *public boolean isWool;* *public run() {} //describe animal run* *}*	AFTER *public class Animals {* *private int age;* *private int weight;* *private int height;* *private String color;* *private boolean isWool;* *private String foodType;* *private run() {}* *public int getAge() { return height;}* *public setfoodType (int newfoodType)* *{foodType= newfoodType}* *}*

Object – oriented programming basics

```
public class Animals
                    {
private int age; //now it is internal attribute
private int weight; //now it is internal attribute
private int height; //now it is internal attribute
private String color; //now it is internal attribute
private boolean isWool; //now it is internal attribute
private run() {} //before any object can made animal run, now animal can decide itself
public int getAge() { return age;} //the way to get information about age
public setFoodType(int newFoodType)  {
                    foodType = newFooodType;} //the way to set new food type for animal
                    }
```

Object – oriented programming basics

Inheritance is relationship between classes when one class repeats structure and behaviour of another class

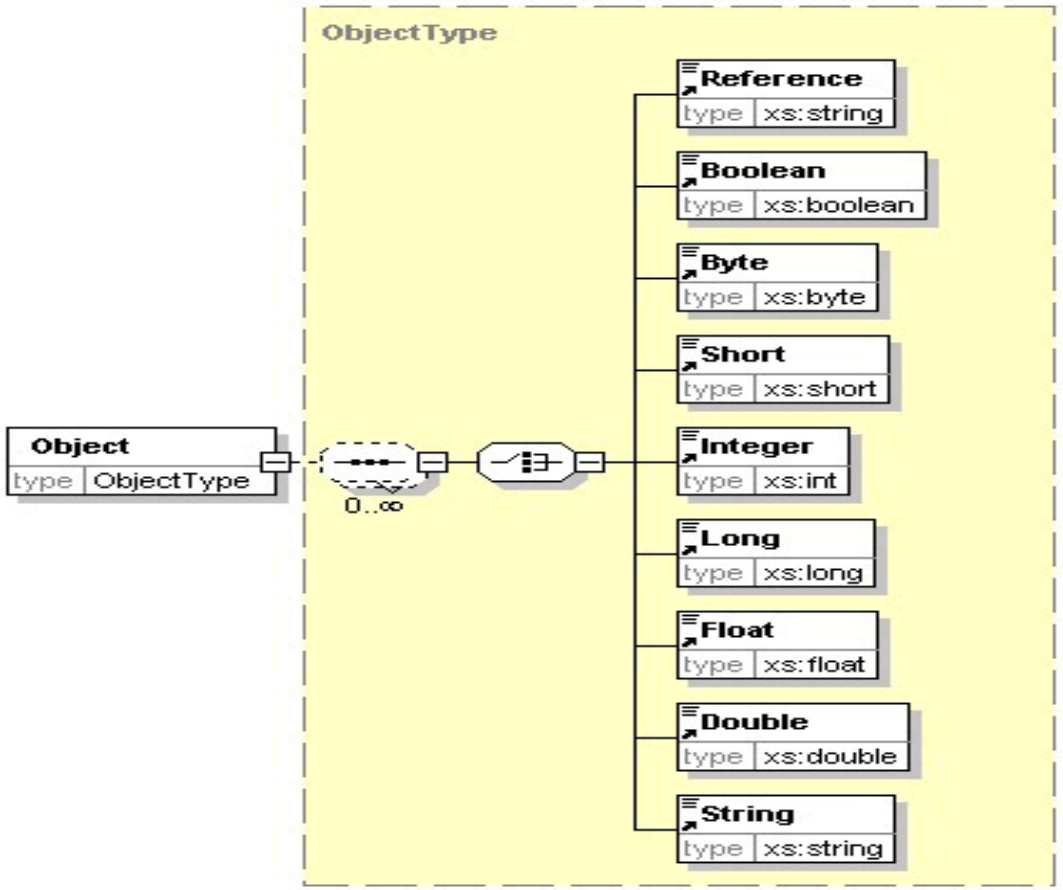
Superclass is a class which structure and behaviour are inherited by another class

Sub-class is a class which inherits structure and behaviour from superclass. It should be mentioned that both structure and behaviour can be re-defined and re-designed

All Java classes inherit Object superclass

Object – oriented programming basics

Class “Object” in Java



Object – oriented programming basics

Initial class	Inherited class

public class Animals {
private int age;
private int weight;
private int height;
private String color;
private boolean isWool;
private String foodType;
private run() {}
public int getAge() { return age;}
public setFoodType (int newFoodType)
{foodType= newFoodType}
}
	public class Dog extends Animals {
	private String breed;
	private String voice;
	private askForWalk() {}
	private askForFood() {}
	public Voice() {}
	}

Object – oriented programming basics

Polymorphism is ability to process different types of data.

In OOP **Polymorphism** is ability of object to use methods of inherited class. This inherited class can de just “on paper” at the moment of object creation.

Polymorphysm term translated from Greek means “Many forms”.

OOP explains polymorphism as “One interface, many implementations”
Sounds complex, right?

Object – oriented programming basics

Example – Web-site with different content

There is web site, and it has three types of content: news, advertisements and articles.

What is same for all three entities: data of publishing, header and content text

What is different for all three entities: articles have authors, news have sources, advertisements have expiration date

Task: display any of three mentioned contents

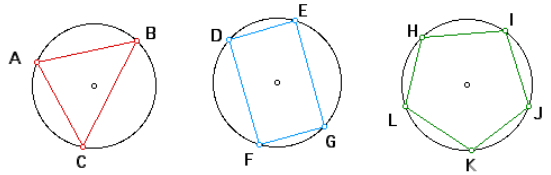
Simple incorrect way: create three methods – showNews(), showArticles(), showAdvertisements()

Complex correct way: create one method – showContent(). This single method should behave differently with different types of content

Object – oriented programming basics

Example – Program that draws geometry figure

There is computer program, and it can draw 5 types of 2-dimensional geometry objects: point, line, triangle, quadrangle and circle.



What is same for all three entities: all objects consist of points

What is different for all three entities: point needs only (x,y) coordinates. Line needs to pairs of coordinates. Triangle and quadrangle – three and four pairs of coordinates accordingly. Circle needs center coordinates

Task: draw requested figure

Simple incorrect way: create 5 methods – drawPoint(), drawLine(), drawTriangle(), drawQuadrangle(), drawCircle()

Complex correct way: create one method – draw(). This single method should behave differently with different types of content

Object – oriented programming basics

So, there are methods, which have the same name, but have to be defined differently for different classes which inherit superclass. Like getContent() and draw().

Basic OOP approach used for it is such:

1) Create superclass, with basic structure, not too much details, but generally containing information about joint attributes and methods of descendants (classes which lately will be inherited from superclass). This class usually can be called **abstract** class. Also if class contains at least one abstract method(which will be re-defined lately in inherited class) – it is also abstract class. There is special modifier **abstract** which states this
2) There is no possibility (OOP rule) to create object of abstract class

Object – oriented programming basics

3) Inherited class has all attributes and methods which belong to abstract class, also it can have its own
4) Method, which is defined differently in inherited class, called **virtual** method
5) Not only abstract classes can be inherited. The case when methods of non-abstract class are re-defined is called **overriding**

Additional info to be read regarding the case

http://docs.oracle.com/javase/tutorial/java/javaOO/classes.html

http://docs.oracle.com/javase/tutorial/java/IandI/abstract.html

Examples

OOP principles presentation http://goo.gl/KKA2WX

Series of video guides from eLearnify project of linda.com

What is object http://goo.gl/qlvTFn

What is class http://goo.gl/RXYiLV

What is abstraction http://goo.gl/5VEnP2

What is encapsulation http://goo.gl/dIlBKr

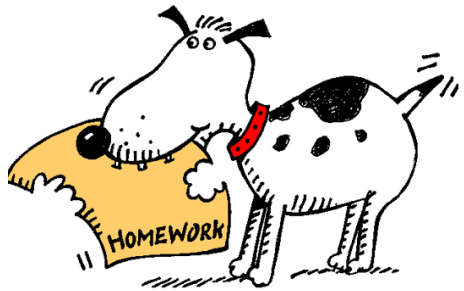
What is inheritance http://goo.gl/qgOUjK

What is polymorphism http://goo.gl/KjZmTF

Homework part 1

1. Install all software required in this presentation. Create file **softwareChecklist.txt** like this http://s3.amazonaws.com/libapps/customers/137/images/checklist.png and push it lately to repository. For all .txt and .java files use UTF8 encoding. Modifications can be done via Notepad++
2. Check if Eclipse is working by following and executing actions in all listed below videos

   https://www.youtube.com/watch?v=xTyizGrqnpI
   https://www.youtube.com/watch?v=LNPIHkOjXs4
   https://www.youtube.com/watch?v=ijT-HRdaKvY



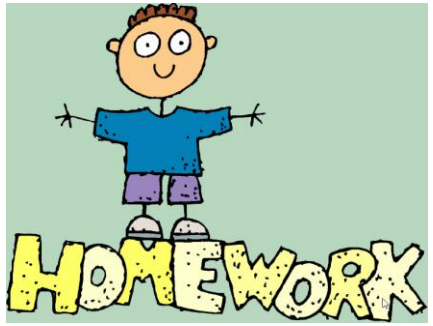
Homework part 1

3. Check if basic operations in GIT (init, commit, push, check out) are working on any 3 dummy files. Delete these files lately.
4. Create file with all needed credentials needed to access your projects, files etc. Name it **credentials.txt**. Create folder **Homework**. Create **Lesson1** in it. Create **Part1** folder. Commit and push created files under your GIT repo there.
5. Provide teacher in separate mail with your GIT repo info (URL, name, login, password)

Homework part 2

1. Create very detailed description of such classes:

- ✓ Customer goods (food and non-food related, usually bought in super markets)
- ✓ Worker of company (think about general classification of all company workers, from cleaner to secretary, from programmer to supplier, from Team Leader to CEO)
- ✓ Touch screen devices (like phones, tablets)
- ✓ Music instruments
- ✓ Entertainments
- ✓ Your own example 1 (don’t use those listed in presentation)
- ✓ Your own example 2 (don’t use those listed in presentation)



Homework part 2

2. Create text-based description together with Java-notation description like in presentation. At least 10 attributes, at least 10 methods (including both private and public).
3. Create separate **[ClassName].txt** files for each class literary description. Create separate **[ClassName].java** files for each class using Java notation.
4. Create folder **Part2** under **Homework1** and push all files there. Note that in Java all class names are started with CAPITAL letter (**Automobile.java**, **Animals.java**, **TestFramework.java**)

Homework part 3

1. Create 4 classes which extend (inherits) EACH of previously created with at least 3 new attributes and 2 new methods. For every class clearly state what is the same and what is different like done in presentation.
2. Create separate **[ClassName].txt** files for each class literary description. Create separate **[ClassName].java** files for each class using Java notation (use example from http://docs.oracle.com/javase/tutorial/java/IandI/abstract.html)
3. Create folder **Part3** under **Homework1** and push all files there.