LEAGUE OF AMAZING PROGRAMMERS

LEVEL 1: CHEAT GUIDE

CREATED BY STUDENTS:

ATHENA HERNANDEZ
CINDY FELDMAN
ELLA DEMAREST
RYLAND BIRCHMEIER
SHIVA KANSAGARA
SOFIA VELARDE

TABLE OF CONTENTS

- A **class** is a template or a blueprint to make objects. The name of a class is usually a noun and starts with uppercase letter.
    - E.g. Smurf.
- Once you create a class, you can make instances of it using the default constructor that Java provides.
- **Instantiate** means to make an instance.
- Example:

```
public class Thing{
    Thing thing = new Thing();
}
```

- **Member variables** give the class the properties that you are interested in.
- A **constructor** (same name as class) will run each time a new instance is made and usually sets up the values of the variables for that instance.

- **Methods** provide actions/functions for instances (unless they are static - see below), so you need an instance to run methods.

- Example**:**

```
public class Person{

    String name;
    int age;

    Person(String name, int age){
        this.name = name;
        this.age = age;
    }

    void printName(){
        System.out.println(”My name is "+ name);
    }
}

Person person = new Person(“Sarah”, 21);
person.printName();    //prints ‘My name is Sarah’
```

- Member variables are normally private so that the class controls how they are processed. This is called encapsulation.

- **Getters and setters** are methods that allow the member variables to be set and accessed.
- A **setter** allows the value of the member variable to be set (so has a void return type).
- A **getter** returns the value of the member variable (so doesn’t have a parameter).

- Example:

```
public void setName(String name){
    this.name = name;
}

public String setName(){
    return name;
}
```

Methods have the following characteristics:

- They have a name; it should be a verb and start with lowercase letter.
- They have parentheses after the name, which hold any parameters. If there is more than 1 parameter, they are separated by commas.
- They have a return type before the name, which defines the type of value that the method returns. If nothing is returned the word ‘void’ is used. You must have a return statement in the method body if a return type has been used.
- They can be public (accessible by other classes in the project) or private (only accessible by their own class).
- The method body is enclosed by mustaches {}.

- **Static** methods and variables belong to the class (and not an instance), so there is only 1 copy and this is shared by all instances.
- The main method is static because this is where the code starts and no instances exist at that point.
- You usually want to break out of the static context (main method) at the start of your program and you can do this using the default constructor to create an instance and run a different method.

```
public class AmazingCode{

    public static void main(String [] args){
        new AmazingCode().go();
    }

    public void go(){
        // amazing code goes here
    }
}
```

- **Swing** is used to make user interfaces. You have to import the relevant libraries at the top of your code to use the different functions.
    - Example: import javax.swing.JFrame;

- Here are some of the most common components:
    - JFrame - provide the basic box for the interface
    - JPanel - allows more than 1 component to be shown at a time
    - JButton - a clickable button
    - JLabel - used to display an image or text
    - JTextField - used to input text

- All of the above are objects and need to be instantiated to use them.
    - Example:

  JFrame frame = new JFrame();
  JPanel panel = new JPanel();

- To make a GUI, you start with the frame and then add things to it.
    - Exampframe.add(panel);

- Normally you need to make the frame visible, allow it close properly and pack the frame.
- <u>Example:</u>
  frame.setVisible(true);
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  frame.pack();

- All of the components have many different methods associated with them, which provide lots of functionality and flexibility.
- Use the instance name and a dot and Eclipse will list all the available methods for you, e.g. frame. will list all the methods for the frame object.

- **Listeners** allow the GUI to 'hear' what the user is doing. There are 3 steps to setting up a listener.
  a. Add the listener to the component:
     e.g.frame.addMouseListener(this);
  b. Allow the class to implement the listener
  c. Add the unimplemented methods

Listeners continued:

- The most common listeners are ActionListener (used to tell if a button has been clicked), KeyListener, MouseListener and MouseMotionListener.
- If the listener detects an event it will run the associated method(s), e.g. if a button is clicked then the actionPerformed method will run. Information about the source of the event is available in the ActionEvent parameter of the method.

- **Binary** is counting using base 2. Each digit is known as a bit, 8 bits make up a byte and the possible values are 0 - 255
  - For example: 01100101
- To convert from binary write in the column values above the digits, starting from 1 on the right hand side and doubling each time.
  128 64 32 16 8 4 2 1
  0   1   1   0 0 1 0 1
- Multiply the column value by the digit and add them together.
  - i.e. 0*128 + 1*64 + 1*32 + 0*16+ 0*8 + 1*4 + 0*2 + 1*1.
- This is the same as adding up all the numbers with a 1 below them. The above example is 64+32+4+1 = 101.
- To convert to binary, try and take away each of the column values starting from 128. If you can put a 1 in that column, if not put a 0. Try and convert 66. You can’t subtract 128, but you can subtract 64 (leaving 2). It starts like this...
  128 64 32 16 8 4 2 1
  0   1
- Then you can only subtract a 2 so you get this...
  128 64 32 16 8 4 2 1
  0   1   0   0 0 0 1 0

- **ASCII** is a code used to translate numbers to text. A has the value 65 and a has the value 97. The binary example above 01100101 (value 101) has the text value of e.

01000111	=	71	=	G
01101111	=	111	=	o
01101111	=	111	=	o
01100100	=	100	=	d
00100000	=	32	=	(space)
01001100	=	76	=	L
01110101	=	117	=	u
01100011	=	99	=	c
01101011	=	107	=	k