1. About Java

- Created by James Gosling.
- Java is an OOP Language.
    - every Java file must contain a class declaration.
    - all code is written inside a class
    - to run a Java program, need to define a main method

1.1 Hello world

```java
public class hello {
    public static void main(String[] args){
        System.out.println("hello world!");
    }
}
```

1.2 Variable

- Java is a statically typed language:
    - all variables, parameters, and methods must have a declared type
    - variables must be declared before use
    - that type can never change
    - expressions also have a type
- Using an IDE, the compiler checks that all the types in your program are compatible before the program ever runs.
- Variables can be declared and initialized once, and assigned multiple times

2. Data Type

2.1 numeric

type	size	range	accuracy
byte	8 bits	-128 to 127 (0 to 255)	
short	16 bits	-32768 to 32767 (0 to 65535)	
int (default)	32 bits	-2^{31} to 2^{31}-1 (0 to 2^{64}-1)	
long	64 bits	-2^{63} to 2^{63}-1	
float	32 bits	+-10^{38} approx.	7 significant digits
double	64 bits	+-10^{308} approx.	15 significant digits

2.2 Boolean

True / False
Its size varies; it only uses 1 bit but can take up to 32 bits of memory.

2.3 Char and String

- string: String are objects and have many methods.
- char: includes non-Roman characters (Chinese), 16 bits.

Methods
char
- `boolean Character.isLetter(c)`：判断是否是字母
- `boolean Character.isDigit(c)`：判断是否是数字
- `boolean Character.isWhitespace(c)`：判断是否是空格
- `boolean isUpperCase(c)`：判断是否是大写字母
- `boolean isLowerCase(c)`
- `char Character.toUpperCase(c)`：转为大写字母
- `char Character.toLowerCase(c)`

string
- `string String.toString(c)`：转为字符串
- `int str.length()`：返回字符串长度
- `str str.substring(int start[, int end])`：左闭右开，取子串
- `boolean str1.equals(str2)`：判断是否相等
- `boolean str1.equalsIgnoreCase()(str2)`
- `boolean str.isEmpty()`：判断是不是空串
- `boolean str1.contains(str2)`：判断是否包含某子串
- `boolean str1.startsWith(str2)`：是否以某子串开头
- `boolean str1.endsWith(str2)`
- `str.toUpperCase()`
- `str.toLowerCase()`
- `str.charAt(int i)` : returns the character at index `i`
- `int str1.indexOf(str2[, int fromIndex])` : it returns the index number where the target string is first found, or -1 if the target is not found
- `int str1.lastIndexOf(str2[, int fromIndex])`
- `string str.trim()`：去掉两头的空格
- `string str.replace(String a, String b)`
- `string str.replaceAll(String a, String b)`
- `string [] str1.split(str2)`

4. Scanner

1. import the package of Scanner

```sql
import java.util.Scanner;
```

3. create a new Scanner object

```sql
Scanner kb = new Scanner(System.in);
```

4. receive and store an input

```sql
String next = kb.nextLine();
```

`next()` 方法是不接收空格的，在接收到有效数据前，所有的空格或者 tab 键等输入被忽略，若有有效数据，则遇到这些键退出。`nextLine()` 可以接收空格或者 tab 键，其输入应该以 enter 键结束。`nextInt()` 和 `nextDouble()` 分别读入相应的类型。

Please DO NOT USE next(), nextInt(), nextDouble(). We will only use nextLine()

5. Casting and Conversion

Primitive Casting

- Casting is explicitly telling Java to make a conversion.
- Casts are required when you want to perform a narrowing conversion. You are instructing the compiler to convert.
- Narrowing runs the risk of losing information.
- The cast tells the compiler that you accept the risk.
- In Java casting is the forcing of conversion between primitive data types, or between objects of different classes.

```sql
y = (datatype)x;
```

Converting between primitive & String

there are classes for ALL primitive data types, they are called wrapper classes. The main use for wrapper classes is to convert text to primitive number types

```java
String s = "999";
int res = Integer.parseInt(s);
String out = Integer.toString(res);
```

6. Flow Controls

6.1 Conditionals

```java
if (condition1) {

}
else if (condition2) {

}
else {

}
```

6.2 Loops

while loop

```java
while (condition) {

}
```

for loop

```java
for (int i=0;i<=10;i++) {

}
```


6.3 Switch

```java
switch (variable) {
case 1:
    break;
case 2:
    break;
default:
}
```

7. Arrays

```java
int [] myArray = new int[5];
```

When created, arrays are automatically initialized
- `String []` : NULL
- `boolean []` : false
- `int []` : 0

initialize the array when declared

```java
int [] myArray = {1, 2, 3, 4, 5};
```

- `myArray.length` : 返回数组长度

8. Method

static method

```java
public static int MethodName(int a){

    return returnValue;
}
```

8.1 Overloading

在同一个类中, 允许存在一个以上的同名函数, 只要他们的参数个数或者参数类型不同即可。

9. Object

~~对象？我没有！~~

```java
public class className {

}
```

9.1 Constructors

构造函数，在实例化一个对象的时候调用并执行。一般用于对对象进行初始化

```java
public class className {
    public className(parameter) {
```

```
    }
}
```

9.2 this

`this` 是一个指向对象的指针，通过 `this.` 的方式来访问对象中的成员变量（属性）

9.3 public and private

- public: 可以在任何地方调用该方法（接口）
- private: 只能在该方法所属的类中调用该方法

9.4 static

static 关键字属于类，而不是类的实例。

静态变量
`static` 可以用于节省内存，所有实例化的对象共用一块静态变量的内存，该静态变量可以被任意对象覆写（有点像全局变量）

静态方法
静态方法不能直接使用非静态数据成员或调用非静态方法，`this` 和 `super` 两个关键字不能在静态上下文中使用。（因为`static` 关键字属于类，而不是类的实例。）

9.5 Inheritance

Java 没有多继承

```java
public class SubClass extends SuperClass {

}
```

A subclass inherits all of the superclass' instance variables, use super() to call the superclass' constructor and pass the initial values.

We can call the inherited public methods on the superclass' instance

9.5.2 Final

If the instance variable will never be changed after it is initialized, we can declare it to be final

```java
private final String str;
```

9.5.2 Redefining Superclass Method in Subclass

在父类中的方法可以在子类中进行覆写

We can put an annotation `@Override` before the overriding method, The purpose is that if we made a mistake such as typo on the method name or parameter then we will get a compile error.

We can still call the overridden method of the superclass by using `super.Method`

9.5.3 Polymorphism

Superclass variable calls the overridden method of each specific subclass

10. Exceptions

Exceptions happen when something goes wrong with the code. A condition that the code cannot handle.

There are two types of Exceptions in Java, checked and unchecked.

checked
Checked exceptions must either be caught by the method in which they occur, or you must declare that the method containing that statement throws the exception. They correspond mainly to file reading and writing. They occurred in compile time, so the program will not run without handling these.

unchecked
Unchecked exceptions are those that belong to subclasses of Runtime-Exception default exception handlers. They can be handled by Java, or you can handle them.

10.1 Try - Catch

把可能抛出异常的代码放入 try，然后使用 catch 来捕获异常，并在块内对其处理

It is possible to have multiple catch blocks to catch different types of exceptions.
`catch(Exception e)` 可以捕获所有异常，它是所有异常的父类，但请不要这么做

10.2 Throw

手动抛出异常

We can use `Exceptions` if
- All return values are valid, and you cannot use a special value to indicate the exceptional case.
- The exceptional case must be dealt with at some early level of method call

Using Exceptions too much will slow down your program.

11. File Input / Output

11.1 File Reading

Step 1: Opening a File

```java
import java.noi.file.Files;
import java.noi.file.Path;

Path path = Paths.get("E:\\data.txt");
BufferedReader reader = Files.newBufferedReader(path);
```

Step 2: Reading Content

```java
String lineContent = reader.readLine();
while (lineContent != null) {
    System.out.println(lineContent);
    lineContent = reader.readLine();
}
```

Step 3: Close File

```java
reader.close();
```

Step 4: Exceptions

```java
Path path = Paths.get("E:\\data.txt");
try {
    BufferedReader reader = Files.newBufferedReader(path);
    String lineContent = reader.readLine();
    while (lineContent != null) {
        System.out.println(lineContent);
```

```
            lineContent = reader.readLine();
        }
        reader.close();
}
catch (IOException ex) {
        System.out.println(ex);
}
```
or
```java
public static void readFile() throws IOException {
        Path path = Paths.get("E:\\data.txt");
        BufferedReader reader = Files.newBufferedReader(path);
        String lineContent = reader.readLine();
        while (lineContent != null) {
            System.out.println(lineContent);
            lineContent = reader.readLine();
        }
        reader.close();
}
```

11.2 File Writing

```java
public static void writeFile(String[] toWrite) {
        Path path = Paths.get("data.txt");
        try {
            BufferedWriter writer = Files.newBufferedWriter(path);
            for (int i=0;i<toWrite.length;i++) {
                writer.write(toWrite[i]);
                writer.newLine();
            }
            writer.close();
        }
        catch (IOException ioe) {
            ioe.printStackTrace();
        }
```

Open Options:
- `StandardOpenOption.APPEND`: new content will be appended to the end of the file
- `StandardOpenOption.CREATE`: create a new file
- `StandardOpenOption.TRUNCATE_EXISTING`: if the file exists, then its length is truncated to 0 (empty the file)

11.3 Creating a Directory

```java
public static void createDir() {
	Path path = Paths.get("newFloder");
	try {
		Files.createDirectory(path);
		System.out.println("Floder: " + path.toAbsolutePath());
	}
	catch (IOException ioe) {
		ioe.printStackTrace();
	}
}
```

12. Graphics

- Graphics: Permits the drawing of shapes, lines and images. The selection of fonts colours and so on
- Components: They are items such as buttons, text fields, menus and scroll bars. We can put them into `containers` and then use `layout managers` to arrange them on the screen
- Containers: Special components that can contain other components
- layout Managers
- Event handlers: handles events such as button clicks, mouse moving etc... Event handlers enable interaction between users the programs

The simplest Window

```java
import javax.swing.JFrame;

public static void main(String[] args) {
	JFrame frame = new JFrame();
	frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	frame.setTitle("My Frame");
	frame.setSize(200, 200);
	frame.setVisible(true);
}
```

12.1 Drawing

Every component inside a container has an x, y, width, and height. The coordinate system of Java GUI
starts at the top-left corner. The (x, y) of a Component is relative to its container.

```java
class Drawing extends JPanel {
	private final int x, y;
	private final int width, height;
	public Drawing(int x, int y, int w, int h) {
		this.x = x;
		this.y = y;
		width = w;
		width = h;
		setBackground(Color.GREEN);
	}
	public int getX() {return x;}
	public int getY() {return y;}
	public int getWidth() {return width;}
	public int getHeight() {return height;}
}
```

The JFrame, will call these methods to get the position, width and height of this ImageDisplay and put it in the relative position within the container. Here we are overriding JPanel's methods.

A number of methods that can be used:
- `g.clearRect(int x, int y, int width, int height)` : Clears the specified rectangle by filling it with the background color of the current drawing surface.
- `g.drawString(String data, int length, int x, int y)` : Draws the text given by the specified character array, using this graphics context's current font and colour
- `g.drawLine(int x1, int y1, int x2, int y2)`
- `g.drawRect(int x, int y, int width, int height)` : Draws a non filled rectangle
- `g.fillOval(int x, int y, int width, int height)` : Draws a filled Oval
- `g.setColor(Color.GREEN)` : Sets the colour to use for drawing

12.2 writing

- `Font myFont = new Font("Serif", Font.BOLD, 12);`
- `g.setFont(myFont)`

12.3 Colours
`Color myColor = new Color(r, b, g);`

12.4 Images

```java
public static BufferedImage readImage(String filename) {
```

```
        BufferedImage image = null;
        try {
            image = ImageIO.read(new File(filename));
        }
        catch (IOException e) {
            System.out.println("read image error");
        }
        return image;
}
```

- `g.drawImage(im, x, y, this)`
- `g.drawImage(im, x, y, width, height, this)`

13. List

13.1 ArrayList

ArrayList cannot hold primitives (int, double, boolean), only objects.

```java
ArrayList<String> lst = new ArrayList<String>();
```


method:
- `add(E e)`
- `add(int index, E e)`
- `contains(E e)`
- `get(int index)`
- `size()`
- `set(int index, E e)` : Replace
- `indexOf(E e)` : First location
- `remove(E e)` : First elemment. returns a boolean to inform if removed successfully.
- `remove(int index)` : return the object

13.2 LinkList

- `LinkList` 继承了 `AbstractSequentialList` 类
- 实现了 `Queue` 接口，可以作为队列使用
- 实现了 `Deque` 接口，可以作为双端队列使用
- 实现了 `List` 接口，可以使用列表的相关操作
- 实现了 `Cloneable` 接口，可以实现克隆
- 实现了 `java.io.Serializable` 接口，可支持序列化

手工链表请看这里：
算法整理 & 复习：链表、双向链表
https://blog.csdn.net/SP_FA/article/details/109405877

14. Time

LocalDate, LocalTime and LocalDDateTime have no time zone.

- `LocalDate`: An instance of `LocalDate` is immutable and represents a date without information of the time and time-zone.
- `LocalTime`: Without information of the date and time-zone.
- `LocalDateTime`: Without time-zone.

```java
LocalDate d1 = LocalDate.now();
LocalDate d2 = LocalDate.of(2021, Month.JULY, 12);
LocalDate d3 = LocalDate.of(2021, 7, 12);
LocalDate d4 = LocalDate.ofYearDay(2021, 193);

LocalTime t1 = LocalTime.now();
LocalTime t2 = LocalTime.of(17, 18);
LocalTime t3 = LocalTime.of(13, 30, 40);
LocalTime t4 = LocalTime.parse("10:15:20");

LocalDateTime dt1 = LocalDateTime.now();
LocalDateTime dt2 = LocalDateTime.of(2021, 7, 12, 12, 30);
```


Period and Duration

- `period`: Represents a quantity of time in terms of years, months, and days.
    - `getYears()`
    - `getMonths()`
    - `getDays()`
- `duration`: Represents a quantity of time in terms of seconds and nanoseconds.
    - `toHoursPart()`
    - `toMinutesPart()`
    - `toSecondsPart()`