Assignment 3

Reminder:

- We provide an online help document for you: 2022春JavaA作业3问题文档, we will update some tips and attentions from time to time.
- You are free to design other fields and methods not mentioned in this document to help you complete this assignment.

For the first 2 problems:

- They are independent and have same submission method as problems of the previous assignment.
- You're encouraged to declare and implement methods as described in the description, but you are **NOT forced** to do that.

For the last 4 problems:

- They are **related to each other** and have different submission approach from ealier ones. You may need to **submit several Java files** according to the submission requirement described in each problem. **DON'T** change the class names to `Main`.
- Please make sure **every field**, including its type and modifiers, **every method**, including its modifiers, parameters (number, type and order), return type are **ABSOLUTELY IDENTICAL** as this document describes.

1. [Easy] Room Escape (20 points)

You are called as the *Ultimate Programmer* (超高校级的程序员). Once you woke up and found that you are stuck in a mysterious room. On the walls there are many strings that you don't know what it means.

A strange voice informs you that the password includes 3 parts of puzzles. You need to figure out all these puzzles to get out of this room! Don't worry, as you are the best programmer, use your super coding ability to solve them!

1.1 Method `getFence()`

```
1 public static String getFence(String cipher)
```

Description

On the wall there is a string which ends with a integer N. You immediately noticed that it is a **Fence cipher(栅栏密码)**. The rule is that **every N letters in this English string should be divided into a group.** Then the group should be printed out in a specific order.

For example: If the input is `HLEOL2`, then `HLEOL` will be divided into 3 groups as `HL`, `EO` and `L`. Then it should take the first letter of each group, then the second letter. By this rearrangement, you will get `HELLO`.

Please implement a method `getFence()` to take in the cipher as input and return the decrypted string.

Sample

- Input Parameter:
  - cipher: `"PCSOEHGDAIO4"`
- Return Value: `"PEACHISGOOD"`

1.2 Method `getCaesar()`

```
1 public static String getCaesar(String cipher, int N)
```

Description

Beside the fence cipher, there are also several strings with a integer N. You noticed that it could be the **Caesar Cipher(凯撒密码)**! The integer N must be how many letters are shifted in alphabet between the cipher and plain.

For example: if the cipher is `ALD` and N is `3`, 3 means that the letter order should be shifted as `A->D`, `L->O` and `D->G`. At last the plain will be `DOG`.

Please implement a method `getCaesar()` to take in the `cipher` and `N` as input and return the decrypted string.

Sample

- Input Parameters:
  - cipher: `Q TJ WQTW`
  - N: `3`
- Return Value: `T WM ZTWZ`

All the plain-texts returned should be an English string without lowercase.

1.3 Method getAnswer()

```
1 public static String getAnswer(String fenceCipher, String caesarCipher, int
caesarN, int M)
```

Description

You are near to victory now, ultimate programmer!

Finally, you discover a piece of paper with a table on it. You know it is definitely **Vigenere Cipher(维吉尼亚密码)**! Also, the paper tells you that the final password should be decrypted by the following way:

- consider the decrypted string by **Caesar Cipher** as the cipher of **Vigenere Cipher**
- consider the first `M` digits of the decrypted string by **Fence Cipher** as the **key** of **Vigenere Cipher**

The table below shows the decryption method using **cipher**, **key** in Vigenere Cipher.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

For example, if the **cipher** is `C`, the **key** is `H`, find where the **row C** and **column H** intersects, which is **J**.

If the **cipher** is `AOPKE`, the **key** is `AB`, expand the length of key by repeating the key, to the length of cipher like `ABABA`. Then, do the corresponding decryption as taught.

Please implement a method `getAnswer()` to take in the origin Fence Cipher `fenceCipher`, the origin Caesar Cipher `caesarCipher`, `caesarN` and the length `M` of the key as input and return the final answer.

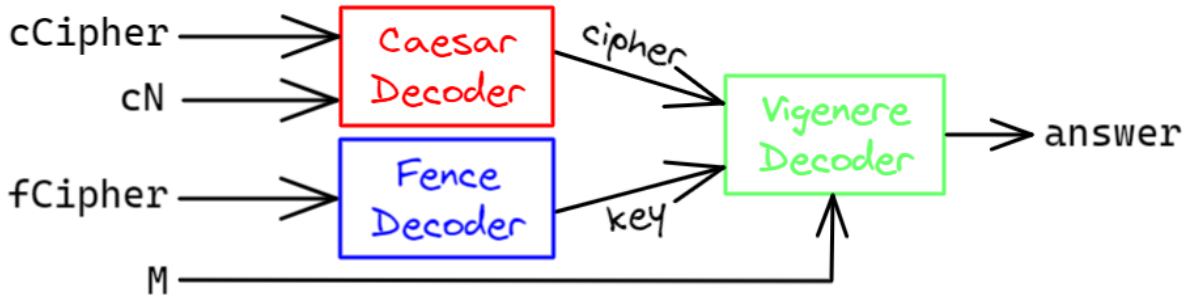
Sample

- Input Parameters:
  - fenceCipher: `PCSOEHGDAIO4`
  - caesarCipher: `Q TJ WQTW`
  - caesarN: `3`
  - M: `5`
- Return Value: `I AM BALD`

All the plain-texts should be returned without lowercase.

1.4 Main Procedure

Now you have implemented 3 super useful ultimate decrypting methods. They will be used in main procedure to solve all the puzzles in this room:



Input Format

First line is the cipher of Fence Cipher `fCipher`

Second line is the cipher of Caesar Cipher `cCipher`

Third line only contains the shift value of Caesar Cipher `CN`

Fourth line only contains the value `M`

Agreement on data range:

- For Fence Cipher, we guarantee that
  - `fCipher` will be a continuous string in only English followed by an integer. $2 \leq fCipher.length \leq 10^6$
  - the groups value N used in this decryption is an integer in range **[1, 9]**, no bigger than the length of cipher-text.
- For Caesar Cipher, we guarantee that
  - `cCipher` will not only contain English. Please only decrypt the English part and keep the others.
  - $1 \leq cCipher.length \leq 10^6$
- For the shift value `N`, we guarantee that $1 \leq N \leq 25$
- For `M`, we guarantee that $1 \leq M \leq fCipher.length$.
- When doing the last decryption, Please only decrypt the English part and keep the others. **When decrypting non-English part, the key will NOT go to the next digit.**

Sample

Sample input

```
PCSOEHGDAIO4
Q TJ WQTW
3
5
```

Sample output

```
I AM BALD
```

2. [Hard] Danganronpa (25 points)

Monokuma leads you to a room through a secret entrance. "This is the class trial field! It's a special place that will decide your fates!" Monokuma says, "Don't be afraid, the trial is not designed for you. Instead, I'd like you to use JAVA helping me solve some problems in the class trial", says the Monokuma.

"Puhuhuhuhuhu... in order to explain my rules more clearly, please look at the brochure!" After saying that, Monokuma gives you the following message:

Hint

1. Large input data, you may need Java Fast IO in https://github.com/Penguin134/CS102A22S12
2. You are likely to encounter **TLE** in this question, which is quite normal. So, you need a more efficient way to solve **TYPE 3** event and the **Final Task**.
3. We provide a solution: hash code (Algorithm: ripemd128)
   `30557199a7502078b04d8ba665a4e5c5`

Problem Description

There will be n events in a class trial. All the events can be classified as *3 types*.

TYPE 1: A participant (excluding you) declare his/her **statement** (a string).

- You'd better storage the **statement** in `ArrayList<> stm` for further uasge.

```
ArrayList<String> stm = new ArrayList<String>();
```

TYPE 2: You blast a **Truth Bullet** (a string).

At the initial state, you have a specific value I of Influence Gauge. In **each event of TYPE 2** you should do
the following operations:

- Storage the **Truth Bullet** in `ArrayList<> tb` for further uasge.
- Your Influence Gauge will increase by `1`.
- Return and print the median (integer) of the length of the **statements** `L`.
  - The medium of $[A_1, A_2, \dots A_i]$ is defined as the $\lceil i/2 \rceil^{th}$ element of $\{A_i\}$ after it is sorted. For example:
    - If `stm = ["ab","abcdef","abcd"]`, then `length = [2,5,4]`. After sorted, `length = [2,4,5]`. Hence, the median `L` is the $2nd$ element of `length`, which is `4`.
  - For special cases:
    - Return and print `0` when no one has put forward any statement.
  - If your Influence Gauge `I` is less than `L`, your Influence Gauge will decrease by the number of
    statements now.

ATTENTION: You should consider ALL the statements here. These steps should be completed in sequence

```
public static int counterStatements(){} // Return the median length of statements
public static void alterInfluenceGauge(int k){} // Alter the value of Influence Gauge
```

TYPE 3: Question Time.

- Return and print how many pairs of `<statement,Truth Bullet>` are matched. Here we define a pair of string `<a,b>` is matched if and only if `a` equals `b` and `a` is provided earlier than `b`. For example:
  - When 1st event happens, `stm = ["Monokuma"],tb = []` there is no matched pair.
  - When 2nd event happens, `stm = ["Monokuma"],tb = ["Monokuma"]` there is one matched pair.
  - When 3rd event happens, `stm = ["Monokuma","Monokuma"],tb = ["Monokuma"]` there is still one matched pair because the second element in `stm` is provided later than first element in `tb`.

ATTENTION: You should consider ALL the statements before the Truth Bullets here.

```
public static int pairMatch(){} // Return the number of pair matched
```

Final Task: After all the event happened, you should tell Monokuma whether you can win the class trial. Here is an equation for you to calculate your final score:

$$Score = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_i b_{n+1-j} m_k$$

Where :
a_i is the return value of `counterStatements()` when event i happens.
b_i is the return value of `pairMatch()` when event i happens.
m_i will be given as the input.

Now, you should do:

- Print `Qi Fei` when $Score \geq 0$
- Print `Fail` when $Score < 0$ OR if your Influence Gauge I drops below 0 in any **TYPE 2** event (no matter the $Score$).

ATTENTION: You should not change the value of Influence Gauge while calculating a_i.

Input Format:

1st line contains two intergers n and I.
2nd line contains n intergers, indicating m_i
3rd ~ (2+n)th line are the events:

- For type 1, each line contains a number `1` and a string s. String s denotes the content of the event.
- For type 2, each line contains a number `2` and a string s. String s denotes the content of the event.
- For type 3, each line contains only one number `3`, denoting the type of event.

Output Format:

For every event, you should follow this output format:
For event TYPE 1, you shouldn't print anything.
For event TYPE 2 and TYPE 3, you should print a number in one line.
And finally you should print whether you win the trial or not.

Agreement on data range:

$$1 \leq n \leq 10^4, 0 \leq I \leq 10^4, |c_i| \leq 10^9, len(s) \leq 100$$

We ensure that there won't be any blank space in a single string.

Samples

Sample input 1

```
3 100
1 1 1
1 ImsureByakuyaduckedunderthetable
2 ImsureByakuyaduckedunderthetable
3
```

Sample output 1

```
32
1
Qi Fei
```

Explanation 1

$$a_1 = a_2 = a_3 = 32 \qquad b_1 = 0, b_2 = b_3 = 1 \qquad Score = 576 \qquad I = 100 + 1 = 101$$

First Line is the return value of `counterStatements()` , second line is the return value of `pairMatch()` .

Sample input 2

```
7 0
2 5 -3 -1 2 1 0
1 No
1 Hemust'veseenthekillertaketheknife
1 ItwassodarkIcouldn'tseemyfood
1 Butitwassuperpitchblack
2 I
2 No
3
```

Sample output 2

```
23
23
1
Fail
```

Explanation 2

$$I_{5th} = 0 + (1 - 4) = -3 < 0$$

Thus, your Influence Gauge becomes less than 0 at the 5th events. You fail the trial.

Sample input 3

```
6 0
1 1 4 5 1 4
1 Thekillerbroughtthemtothecrimescene
1 Ifyoujustusecommonsense
1 Whatiftheglowingpaintwasthemark
1 Inactually,theknifewasfoundunderthetable
3
1 thetablewasbothmarked
```

Sample output 3

```
0
Qi Fei
```

Explanation 3

$$b_1 = \cdots = b_6 = 0 \qquad Score = 0 \qquad I = 0$$

It's okay that you didn't say anything before the end. Maybe that's the so-called win effortlessly 😅.

Question 3-6 Pokémon

> **Designer**: Zhou Junhong, Liu ZhiChen
>
> **Tester**: Nie Qiushi, Wang Biao, Zhang Shaofeng

Description:

Máo máo and Fatherpucci are freshmen at SUSTech. When they were about to experience university life, they found that *the State Press and Publication Administration* had issued a *"Notice on Further Strict Management to Effectively Prevent Minors from Indulging in Online Games"*. Because they are under 18, they are limited to playing games for 3 hours a week. However, they just took the Java course, they plan to write a **Pokémon game** by themselves. The main framework has been set up. Please help them to complete the code together.

3. [Easy] Skill (5 points)

Design a class named `Skill`. It describes the information of a certain skill and will be used in `Battle`.

3.1. Attributes:

- `private String name` : Represents skill's name.
- `private int cd` : Represents skill's Cool Down Time, it will be released when the cd is 0 in the battle.

- `private int atk` : Represents skill's damage while releasing.

3.2. Constructors:

```
public Skill(String name, int cd, int atk)
```

NOTICE:

Please check the validity of the parameters: All `int` type parameters should be **greater than 0**. If any invalid parameter is detected, please **set the attributes to default values**:

```
name = "error"; cd = 51; atk = 0;
```

The detailed **cd** description will be explained in the **battle**.

3.3. Getters:

```
public String getSkillName()
public int getSkillCd()
public int getSkillAtk()
```

In these getters, please **just return the corresponding attributes**.

Notice that we don't need any setters as the attributes are not required to be modified by other classes.

Submission:

For this question, submit one java file: `Skill.java`.

4. [Easy] Pokemon (15 points)

Design a class named `Pokemon`. `Pokemon` will fight in the `battle`.

4.1. Attributes:

`private String name`: Represents Pokémon's name such as “Pikachu”.

`private int hp`: Represents Pokémon's remaining health.

`private int atk`: Represents Pokémon's normal attack damage.

`private int level`: Represents Pokémon's level.

`private int speed`: Represents Pokémon's agility which will determine the order of attack.

`private int rateAtk`: Represents Pokémon's atk increased per level.

`private int rateHp`: Represents Pokémon's hp increased per level.

`Skill skill`: Represents Pokémon's skill.

NOTICE:

To simplify the question, each Pokémon has **only one skill.**

4.2. Constructors:

```
public Pokemon(String name, int hp, int atk, Skill skills, int level, int speed, int rateAtk, int rateHp)
```

NOTICE:

You don't need to check the data input, we guarantee the validity of the data input.

4.3 Setters:

```
public void setHp(int hp)
public void setAtk(int atk)
public void setSpeed(int speed)
```

In these setters, please **just replace the corresponding attribute of the object with the input parameter**.

4.4. Getters:

```
public int getSpeed()
public int getHp()
public int getAtk()
```

In these getters, please **just return the corresponding attributes**.

4.5. Other Methods:

4.5.1. levelUP

```
public void levelUP(int up)
```

This method will be called to increase Pokémon's level by input parameter `up`, which represents the number of levels it will increase. Meanwhile, its `atk` and `hp` will also increase by the functions below:

$$new\,atk = old\,atk + up \cdot rateAtk$$
$$new\,hp = old\,hp + up \cdot rateHp$$

4.5.2. learnSkill

```
public void learnSkill(Skill skill)
```

Pokémon will learn the skill of the input parameter `skill`, this will usually happened during a level up. However, **you are NOT required to judge when to invoke this method** because it will be called manually when we test.

NOTICE:

You are free to design other methods to help you complete this assignment, including but not limited to getters and setters.

Submission

For this question, submit two java files: `Pokemon.java`, `Skill.java`.

5. [Medium] Player (15 points)

You need to write a public class named **Player**, which is used to create player account and store player data (which are pokemons).

(PS: The name of all variables must strictly match the name bolded in the question)

5.1. Attributes:

You are required to write member variables **account, password, mail, phoneNumber** into the class.

- `private final String account`: account number, generated by method `generateAccount()` described in 5.6
- `private String password`
- `private Mail mail`
- `private PhoneNumber phoneNumber`
- `ArrayList<Pokemon> pokemons`: an ArrayList to store Pokémons owned by this player. Don't forget to create an ArrayList object with `new`.

You need to write two additional classes named **Mail** and **PhoneNumber**, each has a **String** to store mail or phone number, and a constructor. What you need to pay attention to is the all members must be **private**, because you need to keep your data safe. However, the member in **Mail** and **PhoneNumber** is not required to be private. Your **Mail** and **PhoneNumber** should be two **classes** declared in the file `Player.java`.

The validity of the structure of mail and phone number are not the point we check, so you don't need to verify the validity of their structure.

5.2. Constructors:

```
1 public Player(Mail mail, String password)
2 public Player(PhoneNumber phoneNumber,String password)
3 public Player(Mail mail, PhoneNumber phoneNumber,String password)
```

You need to write three constructors for each of the three situations for creating an account:

1.Using mail to create an account.

2.Using phone number to create an account.

3.Using both to create an account.

5.3. Checkers:

```
1 public boolean checkIdentity(Mail mail, String password)
2 public boolean checkIdentity(PhoneNumber phoneNumber, String password)
```

You need to write two methods to check the identity of player when he/she logs in. Only when the data input is match the data you stored means the player passes it successfully. If succeed return true, else return false.

5.4. Setters:

```
public boolean setMail(PhoneNumber phoneNumber, String password, Mail mail)
public boolean setPhoneNumber(Mail mail, String password, PhoneNumber
phoneNumber)
```

You need to write two methods to set (or to change) the mail or phone number. You need to **check the player's identity first**, after that you can change the data. If succeed return true, else return false.

5.5. Getters:

```
public String getAccount()
public Mail getMail()
public PhoneNumber getPhoneNumber()
```

In these getters, please **just return the corresponding attributes**.

5.6. Account Generator:

You need to write a method that can generate account automatically. The account must be a random number string with length 7, and the first number of it can't be 0.

```
public String generateAccount()
```

Attention:

1. This method should be invoked when you create an account. The return value should be assigned to the attribute `account`.
2. Once your account is created, it can't be changed anymore.

5.7. Change Password:

```
public boolean changePassword(PhoneNumber phoneNumber, String oldPassword,
String newPassword)
public boolean changePassword(Mail mail,String oldPassword, String
newPassword)
```

You need to write two method to change your password. You need to **check the player's identity first**, after that you can change the password. If succeed return true, else return false.

5.8. Add Pokemon:

```
public void addPokemon(Pokemon pokemon)
```

You need to write a method to add Pokémon to the `ArrayList<Pokemon> pokemons`. We ensure the input Pokémon object is valid.

Submission:

For this question, submit three java files: `Player.java`, `Pokemon.java`, `Skill.java`.

PS: Your **Mail** and **PhoneNumber** should be two **classes** declared in the file `Player.java`.

6. [Hard] Battle (20 points)

After finished the class **Player** and **Pokemon**, you finally can write a battle system. You are required to write a class named **Battle**, which contains only one **static** method named **tatakai**.

6.1 tatakai:

```
1 public static Player tatakai(Player p1, Player p2)
```

The process of a battle is as below. I know this should be a hard time, but please read the following text carefully:

Process

The round limit is **50**.

First you need to input two players p1 and p2. Then you should get the first Pokémon from the two players' pokemon list seperately. Compare these two Pokémons' speed, **the faster one will start first (if same, then p1 start first)**. After these preparations, the battle starts.

Pokémons attack each other in each turn. The rule of generating damages is: `pok2.hp -= pok1.atk;` `pok1.hp -= pok2.atk;`

If Pokémon has skill, it will use its skill in the battle when the cd is reached. **The cd starts calculating at the moment when the Pokémon get into the stage, when the cd is reached, Pokémon will replace its normal attack by skill. The skill can be used infinite times. When the skill is released, it start to count cd again.**

If one Pokémon's hp comes to 0, **this round over and this Pokémon will quit the stage**. The Pokémon whose hp is not 0 will stay on the stage. **The player who lost his Pokémon should let his next Pokémon get into the stage, and do speed judge again to decide which side starts first, and repeat the battle above**, until one player loses all his Pokémons, or the round has reach the limit, the game is over.

When the game is over, the one who still have Pokémon is the winner, you should return this player. If the round has reach the limit (**at the end of round 50**) and no one lose, the game ends in a draw, and you should return null instead of player.

Also, before return statement, the `hp` of all the Pokémons should be **refilled to its initial value.**

NOTICE:

- The round you record over the **upper limit of the number of rounds** and the round you used to calculate **when to release the skill** are **not** the same round.
- We ensure the two Players both have **at least one Pokémon** in the list `pokemons` when tatakai.

Submission:

For this question, submit four java files: `Battle.java`, `Player.java`, `Pokemon.java`, `Skill.java`.

Test Procedure Sample

```
player1 = new Player(new Mail("1@mail.sustech.edu.cn"), new PhoneNumber("1"), "1");
player2 = new Player(new Mail("2@mail.sustech.edu.cn"),new PhoneNumber("2"), "2");
Skill skill1 = new Skill("skill1", 2, 3);
Skill skill2 = new Skill("skill2", 3, 2);
Pokemon pokemon1 = new Pokemon("pokemon1", 10, 1, skill1, 1, 1, 3, 3);
Pokemon pokemon2 = new Pokemon("pokemon2", 10, 1, skill2, 1, 2, 3, 3);
player1.addPokemon(pokemon1);
player2.addPokemon(pokemon2);

Player winner = Battle.tatakai(player1, player2);
```

Since `pokemon1.speed < pokemon2.speed`, pokemon2 attack first.

Below shows what happens in each round:

```
-- Round 1 --
pokemon1.hp = 10; pokemon2.hp = 10;
pokemon2 attack first, its attack value is normal attack damage: 1;
pokemon1.hp = 9;
Then, it's pokemon1's turn to attack, the attack value is normal attack damage: 1;
pokemon2.hp = 9;

-- Round 2 --
pokemon1.hp = 9; pokemon2.hp = 9;
pokemon2 attack first, its attack value is normal attack damage: 1;
pokemon1.hp = 8;
Then pokemon1's skill is ready, it's attack value is skill damage: 3;
pokemon2.hp = 6;

-- Round 3 --
pokemon1.hp = 8; pokemon2.hp = 6;
pokemon2's skill is ready, its attack value is skill damage: 2;
pokemon1.hp = 6;
Then, it's pokemon1's turn to attack, the attack value is normal attack damage: 1;
pokemon2.hp = 5;

-- Round 4 --
pokemon1.hp = 6; pokemon2.hp = 5;
pokemon2 attack first, its attack value is normal attack damage: 1;
pokemon1.hp = 5;
Then pokemon1's skill is ready, it's attack value is skill damage: 3;
pokemon2.hp = 2;

-- Round 5 --
pokemon1.hp = 5; pokemon2.hp = 2;
pokemon2 attack first, its attack value is normal attack damage: 1;
```

```
pokemon1.hp = 4;
Then, it's pokemon1's turn to attack, the attack value is normal attack damage: 1;
pokemon2.hp = 1;

-- Round 6 --
pokemon1.hp = 4; pokemon2.hp = 1;
pokemon2's skill is ready, its attack value is skill damage: 2;
pokemon1.hp = 2;
Then pokemon1's skill is ready, it's attack value is skill damage: 3;
pokemon2.hp = -2;

Since pokemon2.hp < 0, pokemon1 win. The winner is player1.
```