Class Creation

AP Computer Science A · Vocabulary in context

Word	Example
Abstraction /əbˈstrækʃn/	**Abstraction** hides details you do not need.
interface /ˈɪntəfeɪs/	The **interface** lists the methods other code can use.
readable	Clear names make code **readable**.
reusable /riːˈjuːzəbl/	A well-designed class is **reusable** in other programs.
Encapsulation /ɪnˌkæpsjʊˈleɪʃn/	**Encapsulation** keeps an object's data private.
private /ˈpraɪvət/	Use **private** fields to hide the stored balance.
methods /ˈmeθədz/	A car's **methods** include Drive() and Stop(). Drive() Stop()
state /steɪt/	An object's **state** is the values of its fields.
testing /ˈtestɪŋ/	Use **testing** to compare actual and expected answers.
bugs	Shadowed variables are a common source of **bugs**.
System reliability /ˈsɪstəm rɪˌlaɪəˈbɪlɪti/	**System reliability** means the program keeps working as expected.
beneficial /ˌbenɪˈfɪʃl/	Programs can be **beneficial**, for example by speeding up medical tests.
harmful /ˈhɑːmfl/	Programs can also be **harmful**, for example by leaking personal data.
legal /ˈliːgl/	Firms must follow the **legal** rules of each country.

exercises.lol exercises.com.cn

Word	Example
intellectual-property	Code is **intellectual-property**, protected like a book.
open source /ˈəʊpən sɔːs/	The **open source** licence lets us read and change the code.
licence /ˈlaɪsəns/	You need a **licence** to sell alcohol in a shop.
give credit	Always **give credit** when you reuse someone else's code.
instance variables /ˈɪnstəns ˈveərɪəblz/	**Instance variables** store each object's own data.
fields	The **fields** hold an object's data.
behavior /bɪˈheɪvjə/	An object's **behavior** is what its methods do.
constructor /kənˈstrʌktə/	The **constructor** sets up a new object.
return type	A constructor has no **return type**, not even void. void
initialize	The constructor should **initialize** the player's starting score.
parameter lists	Overloaded constructors have different **parameter lists**.
overloading /ˌəʊvəˈləʊdɪŋ/	**Overloading** means two methods have the same name but different parameters.
no-argument	A **no-argument** constructor takes no parameters.
defaults	Java's **defaults** include 0 for int instance fields. Java int 0

Word	Example
signature /ˈsɪgnɪtʃə/	A method's **signature** is its name and parameter types.
body /ˈbɒdi/	The method **body** is the code between the braces.
accessor (getter) /əkˈsesə/	An **accessor (getter)** returns a field's value.
mutator (setter) /mjuːˈteɪtə/	A **mutator (setter)** changes a field's value.
reference /ˈrefrəns/	A variable holds a **reference** to an object, not a copy.
caller	A change to a shared object is visible to the **caller**.
Primitives /ˈprɪmɪtɪvz/	**Primitives** such as int and double store values directly. int double
copied by value	Arguments are **copied by value** in Java. Java
immutable /ɪˈmjuːtəbl/	Strings are **immutable**: you cannot change one letter in place.
mutable	An ArrayList is **mutable**: a method can add to it. ArrayList
static (class) variable /ˈstætɪk ˈveərɪəbl/	A **static (class) variable** is shared by all objects of the class.
static method /ˈstætɪk ˈmeθəd/	A **static method** is called with the class name.
Scope /skəʊp/	**Scope** decides where a variable can be used.
local variable /ˈləʊkl ˈveərɪəbl/	A **local variable** is used only inside its procedure.
parameter /pəˈræmɪtə/	In def area(r):, r is a **parameter**. def area(r): r

exercises.lol exercises.com.cn

Word	Example
Access modifiers	**Access modifiers** such as private and public control visibility. private public
visibility	A public method has wider **visibility** than a private method. public private
shadow /ˈʃædəʊ/	A parameter with the same name will **shadow** the field.
current object	this refers to the **current object**. this
base case /beɪs keɪs/	A recursive method stops at its **base case**.
recursive call /rɪˈkɜːsɪv kɔːl/	Each **recursive call** works on a smaller problem.
inheritance /ɪnˈherɪtəns/	With **inheritance**, a Bus class gets the methods of the Vehicle class. Bus Vehicle
superclass /ˈsuːpəklæs/	Vehicle is the **superclass** of Bus. Vehicle Bus
subclass /ˈsʌbklæs/	Bus is a **subclass** of Vehicle. Bus Vehicle
method overriding /ˈmeθəd ˌəʊvəˈraɪdɪŋ/	**Method overriding** lets a subclass change an inherited method.