# Homework 6

**Q1:** There are four types of access modifiers in Java

**Private:** Methods, variables and constructors accessible only with the declared class. This means we cannot reach these members with any other class (include class of same package). When we are accessing the private members from outside the class compile error occurs.

**Default (package-private):** If you don't declare any modifier, it is handled as default. Any variable, method or classes are accessible on the *package* that belongs. This means we cannot reach from the outside of the package.

**Protected:** This access modifier is accessible within *same package* or with *subclasses* in different package. Protected can be applied on the data member, method and constructor. Protected modifier cannot applied to a class and interfaces. Methods, data fields can declared with it.

**Public:** Public access modifier is accessible from everywhere. It has the widest scope among all other access modifiers. There is no restriction on the scope of public data fields, class, method constructor.

According to the explanations above private and default access modifiers has more restrictive than protected and public. The answer is the protected access modifier allows access to everything the package-private modifier does and more.

**Answer is Option C**

**Q2:** A **constructor** is a special method that called when create a new object. The constructor create objects and give their instance variables with default initial values. Each constructor has the name the same of class name. In addition, constructors does not have a return type. If you write a class with no constructors, Java provides no argument constructor (default) for you. There is a constructor class in java and it used to get internal information on a constructor in the class.

Method **this()** used as a reference to the members of a class just like constructors, variables and methods. **super** is a keyword to refer parent class method or instance. Super() method calls the parent class constructor with no arguments. There aren't any **that()** and **construct()** methods in java.

**Answer is Option B**

**Q3:** The sell() method has a **boolean** type. This means that this method returns a boolean value **(true, false).** There are several return statements in code but they are placed in **if** and **else if** condition blocks. After the scopes this code does not return any boolean value. It need an return expression. So **t**his code does not compile.

**Answer is Option D**

**Q4:** At the heading of the method that describe the return type. If method does return any value, it takes the **void**. When the code trying to run, it gives the error message. The message includes the print function does not applicable for the arguments. It means that method's (pointed with g1) **return type** is undefined. Thus, it does not compile because of line **g1**

**Answer is Option B**

**Q5:** There are two ways to passing parameters in programming languages. One of them **pass (parameters) by reference.** This defined by formal parameter is just an alias of the actual parameter. It refers to actual argument. Any changes done to formal argument will affect in actual argument. The other way is **pass (parameters) by value.** This means that copy of the value will passed to a method. The Java spec says that everything in Java is **pass by value**

There is no difference between passing primitive data types and Objects when we talking about method arguments. You always copy of the bits of the value of the reference. If it is a **primitive data type** these bits will contain the value of the argument **(primitive data type) itself.** Else it is a **Class type variable** (Object) the bits will contain the reference's value (memory address) of the object argument is copied to the parameter.

There is a link about the pass by value / reference discussion below:
**https://stackoverflow.com/questions/40480/is-java-pass-by-reference-or-pass-by-value**

**Answer is Option B**

**Q6:** For constitution of encapsulation, the data is private and getter/setter methods are public. This naming convention is used in JavaBeans. JavaBeans are reusable software components. Its used for allows access to properties using getter and setter methods.. There are some syntactic rules regard as JavaBean class.

- It should be **public** modifier. And written with semantic manner.
- The return type should be **void**
- It should be **serializable**. It can implement the Serializable interface.
- The set method prefixed with **set / get** for getter setter methods.
- It should take some argument (for setter methods). No argument methods used (for getter method)

**Answer is Option C**

**Q7:** **Constructor chaining** is the process of calling one constructor from another constructor with respect to current object. This chaining process occurs through inheritance. A constructor can have either **this or super** keyword but not **both**. In java another constructor of the same class can be called from a constructor via **this()**. This has to be on first line. The call to **super()** must be the first call in the constructor. There should be at least one constructor to write a constructor without the **this** keyword.

**Answer is Option B**

**Q8:** Each method is specified in the type it will return in java. 10 is an integer and if the method returns a integer value. It must declared with integer type

**Answer is Option A**

**Q9:** **Public** variable is always available to all instances of the class.

**Local** variables are available to declared method, constructor or block

**Static** variables are visible that a class belongs.

**Instance** variables are visible for all methods, constructors and a block in the class.

**Answer is Option A**

**Q10:** If we inserted **this(4) boolean** outside has a default value **false** it enters the ternary statement and incremented to the five

**Answer is Option A**

**Q11:** An instance of one class may **not** access **package-private** attributes in a parent class, provided the parent class, provided the parent class is not in the same package. It need to be **protected**. It is false

**Answer is Option B**

**Q12:** There are some guidelines for defining a well-encapsulated class.

- Declare all the instance variables in the class with **private.**
- Provide **public getter** method to retrieve the data an object and **setter** method for manipulating the data in the class.
- Make any helping methods private.
- Write comments within the class definition to describe implementation details.

In this question we are eliminated the first sentence. The other solutions are invalid.

**Answer is Option D**

**Q13:** If a class contains no constructor declarations, then a default constructor with no formal parameters and no throws clause is implicitly declared by Java. It contains a default call to **super** class constructor. If you implement any constructor, you have no default constructor.

**Answer is Option A**

**Q14:** When using the varargs there are only one variable argument in the method. Another rule is variable argument must be the last arguments. Only option A meet the varargs requirements.
**Answer is Option A**

**Q15:** After the calling slalom() method mySkier object has age value 18 taken from instance. In other words, it has Ski type object and has a 18 a value mySkier points it. The name is Rosie because "Wendy" is a local variable. We cannot reach from the outer scope of class. An array has a default value zero. The new created array is takes the value from main method there is not any new value inserted in. This conditions showed in diagram in option C.

**Answer is Option C**

**Q16:** Method overloading has **same** name but **different** usage of parameters. Method overloading is not possible by changing the return type of the method only. It cause the ambiguity. There is an compile error occurs. There are two ways of **overload** the method in java. One of them is changing the number of arguments and the other is changing the data type. Best representation of overloading is in option B.

**Answer is Option B**

**Q17:** Data encapsulation allows programmers to enforce class constants and conditions. When this data files marked with the private, no one sets them to illegal values. This provides us the **data integrity.** A private is also hides the program pieces from the clients. So they cannot try to modifying the **attributes** of class. Separate implementation allows the interface independently. Encapsulation also improves the **reusability** and easy to change with new requirements. Encapsulation does not affect the program performance.

**Answer is Option D**

**Q18:** We can use any data type for a parameter of a method or constructor. It includes primitive data types and reference data types such as objects and strings. **Primitive type arguments** are passed into methods by value. This means that any changes of the values of parameters exist only within the scope of the method. Any differences are gone after the method execution. **Reference data types** are also passed into methods by value. When the method returns, the passes in reference still references the same object as before. However the values of object fields can modified. In this question string is only object type data type.

**Answer is Option B**

**Q19:** In this question, we want to call from another class in the same package. The method is public and we can reach everywhere. We can calling with

```
ClassName.MethodName(Arguments)
```

**Answer is Option D**

**Q20:** If method does not need to return any type, it written with the void type.

**Answer is Option C**

**Q21:** In this application there are **final int score** as an argument. The compiler knows that a method can called multiple times and parameter used too. So the compiler directs to programmer assign a value to a final variable or remove it. Same thing is available in the result variable. When the method is called new values come to the result variable.

**Answer is Option C**

**Q22:** **Super** keyword used for access the data member of parent class. **Super()** method is used to calling the parent class constructor. So the answer is super() super.

**Answer is Option D**

**Q23:** If the code has not include any access modifier, it applied as default by compiler. Default access modifier is a package-private modifier. Any variable method or lasses are accessible classes within same package.

**Answer is Option B**

**Q24:** To encapsulating properly, all instance variables become **private** access modifier. Then adding of the getter and setter methods to the data field is needed. According to the explanations, all of the statements needed.

**Answer is Option C**

**Q25:** An method names may only include letters, numbers, $ (dollar sign) and _ (underscore). Also the first character is not allowed to be a number. Moreover, reserved words are not allowed. Method name placed after the return type. A method can start with capital letter and end with punctuation. Also there is a name of the method. Only **new()** method is valid.

**Answer is Option C**

**Q26:** Given code, there is a mistyping error here. The compile accepts when the integer type as placed before the method name. Since the method type integer it will return an interger. But immediate situation code does not compile.

**Answer is Option D**

**Q27:** Java use the pass by value so we cannot pass in primitive and modify the primitive value. However we can return the value from the method and reassign it. You could create an object that allowed a method to get and set the int value, you just modify the object itself

**Answer is Option B**

**Q28:** The final object type content needs to be initialize. It can provided by deleting the final keyword. Unless there is a compile error. The result is code cannot compiled.

**Answer is Option C**

**Q29:** Some valid JavaBean method prefixes are **is , get, set** other words are common words.

**Answer is Option A**

**Q30:** If we want a reach the class that it belongs to other package, **import** keyword used. When the keyword is used statement import word must be at first. The getClothes method is a static method, since we can add static keyword into the import statement. The accessing phase start where the class found the packages. After the package name wrote the class name (Store) then end with the method name (getClothes) The result is showed in option C

**Answer is Option C**

**Q31:** If there is no access modifier definition, it goes the package-private. Default is not a type of access modifier.
**Answer is Option D**

**Q32:** There is only one line create compile error its includes super() keyword. The error message says that super statement must be first statement in a constructor.

**Answer is Option B**

**Q33:** Constructors can't become **final**, **static** or **abstract** in Java. Static method used for some code that easily shared by all the instance methods. Static method can only access static attributes. Instance method can access static variable and static methods directly. Static methods can't access instance methods and instance variables directly. They must use reference to object.

**Answer is Option A**

**Q34:** In this question, a method type cannot change to another return type. The most accaeptable option is d because of the same data type.

**Answer is Option D**

**Q35:** Overloaded method must have same name, there has a different list of parameters. Overloaded methods only allowed for different signatures. The return type not included in method signature. So overloaded methods can differ with the data type.

**Answer is Option C**

**Q36:** In these lines, there is only declaration of different data fields. None of them related to each other. The will not compiled anyway.

**Answer is Option D**

**Q37:** This code compiles when the remove the 2 parameter in last line. This code not compiled because of line q3.

**Answer is Option D**

**Q38:** The public access modifier allows access to everything the private access modifier does an more.

**Answer is Option A**

**Q39:** This code takes the three as a final parameter and return it to the output.

**Answer is Option A**

**Q40:** There is a water method defined and defines the constructor with the same name of a method name. There is not many method that need of use this keyword.

**Answer is Option A**

**Q41:** The method arguments has the compatible type of data type method declaration. There is an int String String definition in call method. This is a match with option C.

**Answer is Option C**

**Q42:** Static method used for some code that easily shared by all the instance methods. Static method can only access static attributes. Instance method can access static variable and static methods directly. Static methods can’t access instance methods and instance variables directly.

**Answer is Option B**

**Q43:** A class does not have a constructor explicitly defined. If there is no constructor in a class Java always create a non argument constructor as a default.

**Answer is Option B**

**Q44:** This code does not compile because there are two of the static expression and height has an ambiguity about it.

**Answer is Option D**

**Q45:** The code does not compiled because there is not a super class definition in given code

**Answer is Option D**

**Q46:** The code does not compiled because there is not any method declaration about choose method.

**Answer is Option D**

**Q47:** The code does not compiled because the method is looking for int not long data type.

**Answer is Option C**

**Q48:** Method names may only include letters, numbers, $ (dollar sign) and _ (underscore). Also the first character is not allowed to be a number. Moreover, reserved words are not allowed. Method name placed after the return type. A method can start with capital letter and end with punctuation. Also there is a name of the method. Only **$sprint()** method is valid.

**Answer is Option A**

**Q49:** **Protected** access modifier is accessible within *same package* or with *subclasses* in different package. Protected can be applied on the data member, method and constructor. Protected modifier cannot applied to a class and interfaces. Methods, data fields can declared with it. The option B meets the requirements.

**Answer is Option B**

**Q50:** None of the import statements not compiled well.

**Answer is Option D**