"Approved"
Chairman of the Academic
and Methodological Council
_____ Abdreshov Sh.A.
"20" 11 2025

LIST
of questions and practical tasks
for the attestation (comprehensive) examination
for the educational program
"6B06118 – Software Engineering"

1. Discipline "Software Development Tools"

1. Integrated Development Environments (IDEs): Purpose, Features, and Benefits.
2. Architecture and features of using Visual Studio, PyCharm and VS Code.
3. Version control systems: how it works, basic Git commands.
4. GitHub, GitLab, and Bitbucket: Feature comparison and use case.
5. Branching processes in Git (branching models): Git Flow, Feature Branches.
6. Build automation tools: Maven, Gradle, Make.
7. Apache Maven project management system: project structure and POM file.
8. Package managers (npm, pip, NuGet): purpose and features of work.
9. Dependency Management in Software Projects: Principles and Tools.
10. Code documentation tools (Sphinx, Javadoc, Swagger).
11. Software testing tools: JUnit, PyTest, NUnit.
12. Types of testing and test automation.
13. Debugging programs: tools, methods, breakpoints, viewing variables.
14. Application logging and monitoring: tools and best practices.
15. Docker: Containerization basics and creating a Dockerfile.
16. Docker Compose: Multi-Container Application Management.
17. Containerization vs virtualization: comparison and areas of application.
18. Continuous Integration and Delivery Systems (CI/CD): Principles and Pipelines.
19. GitHub Actions, GitLab CI, Jenkins: comparative characteristics.
20. Software version control and releases.
21. Code quality analysis tools: SonarQube, ESLint, Pylint.
22. UML modeling: tools (Visual Paradigm, StarUML, Enterprise Architect).
23. CASE tools and their role in the development of software systems.
24. Figma and UI/UX prototyping tools in software development.
25. Team collaboration tools: Jira, Trello, Asana.
26. DevOps approach: basic tools and stages of the life cycle.
27. Configuration management tools: Ansible, Puppet, Chef.
28. Virtual development environments: venv, Conda, VirtualBox.
29. Cloud development tools: AWS, Azure, Google Cloud.
30. Code security control tools and static analysis (SAST/DAST).

2. Discipline "Fundamentals of Computer Modeling"

1. The main features of the Python language are interpretation, dynamic typing, and applications.
2. The structure of the program is in Python. Variables, data types, and operations.
3. Control constructs: conditions (if/elif/else) and loops (for, while).
4. Strings and Row Operations. String methods.
5. Lists in Python: Creating, Indexing, Slicers, Basic Methods.

6. Tuples and their features. Differences between tuples and lists.
7. Dictionaries and sets: data structure, methods, areas of application.
8. Functions: declaration, parameters, scopes, recursion.
9. Anonymous functions (lambda), higher-order functions (map, filter, reduce).
10. Working with files: reading, writing, file modes.
11. Exceptions: error handling, try/except/finally blocks.
12. Modules and packages: import, organization of the project structure.
13. Python virtual environments (venv, conda).
14. Object-oriented programming: classes, objects, methods.
15. Inheritance, encapsulation, polymorphism.
16. Dunder methods and operator overloading.
17. Generators and iterators.
18. Decorators: purpose, principles of work, examples.
19. Work with dates, times, and the datetime library.
20. Working with databases in Python (sqlite3, SQLAlchemy).
21. Fundamentals of Data Processing: NumPy Library.
22. Tabular data and data analysis in the pandas library.
23. Data visualization: matplotlib, seaborn.
24. Basics of network programming: sockets, HTTP requests.
25. Basics of multithreading and multiprocessing.
26. Work with API, JSON, and XML.
27. Python web development basics: Flask/Django (structure, routes).
28. Python code testing: unittest, pytest.
29. Logging and debugging of programs (logging, pdb).
30. Practices for writing high-quality Python code: PEP 8, typing.

3. Discipline "Object-Oriented Programming"

1. The basic principles of object-oriented programming are encapsulation, inheritance, polymorphism, and abstraction.
2. The concept of class and object. Class structure, properties (attributes), and methods.
3. Designers and destructors. Object life cycle.
4. Access modifiers: public, private, protected (in C++/Java languages).
5. Encapsulation: definition, benefits, practical examples.
6. Inheritance: Types of inheritance, class hierarchies, method overrides.
7. Polymorphism: static and dynamic. Examples of implementation.
8. Abstract classes and interfaces. Application in system design.
9. Override and overload methods.
10. UML Class Diagrams: Elements, Relationships, Design Use.
11. Composition and aggregation: differences and examples.
12. Multiple inheritance: benefits, challenges, and solutions.
13. Exceptions in OOP: error handling, exception hierarchies.
14. Static Class Members: Static Methods and Variables.
15. Virtual functions (in C++), dynamic linking mechanisms.
16. Object generation and memory management (new/delete, garbage collector).
17. Design Patterns: Role and Classification.
18. Generative patterns: Singleton, Factory, Abstract Factory.
19. Structural patterns: Adapter, Decorator, Composite.
20. Behavioral patterns: Strategy, Observer, Command.
21. SOLID Principles: Description and Application in OOP.
22. Work with collections of objects: lists, dictionaries, sets (in different languages).
23. Use generics in Java/C# and template in C++.
24. Operator overloading: purpose and examples.

25. Interfaces and how they differ from abstract classes.
26. Object serialization mechanisms.
27. Event and subscriber processing (event-driven programming).
28. A system of modules and namespaces (namespace, package).
29. Principles of object-oriented code refactoring.
30. The use of OOP in modern languages: Java, C#, Python, C++, Swift.

4. Tasks

1. Analysis of the sales dataset (CSV + Pandas + charts)
Upload a CSV file (any: sales, students, transactions).
Fulfill:
- output of the first lines;
- Data types
- Missing values
- Top 5 best-selling categories;
- build a sales graph by months (line plot).

2. Statistical analysis of the data set (NumPy + visualization)
Generate a random array of 1000 numbers.
Calculate:
- mean, median, variance, standard deviation;
- build a histogram of distribution;
- Find outliers (values > 3σ).

3. Text analysis (NLP basics + Counter)
Upload a text file (or take text from the Internet).
Fulfill:
- cleaning the text from punctuation;
- word counting;
- Frequency analysis of words (Counter);
- Visualization of the top 20 words (bar chart).

4. Working with GitHub API (requests)
Get GitHub User Data:
- login, followers, public repos;
- creation dates;
- List of repository names
- Display the top 5 repositories by number of stars.

5. Analysis of film data (pandas + grouping)
Use a ready-made IMDB or TMDb dataset (available in the public domain).
Fulfill:
- top 10 films by rating;
- average rating by genre;
- visualization: rating vs budget;
- Find movies with a payback > 200%.

6. 3D Data Visualization (Matplotlib 3D)
Generate 3D points:
- x, y — random;
- z is the function $z = \sin(x) + \cos(y)$;
- Build a 3D graph of the surface or scatter plot.

7. Linear regression model (sklearn)
Generate data:
- x — random values;
- y = 3x + 7 + noise.

Fulfill:
- LinearRegression model training
- visualization of the regression line;
- MAE, MSE, R2.

8. Working with a high-level nested JSON file
Upload JSON (e.g., weather data, GitHub API, or your own).
Tasks:
- withdraw all level 1 keys;
- Find all items of the "list" type.
- extract specific fields (temperature, date, location);
- convert to a DataFrame.

9. Time Series Analysis (pandas + resample)
Use a dataset with dates (for example, cryptocurrency prices, exchange rate).
To be done:
- converting the index to datetime;
- aggregation by day/month;
- moving average;
- trend chart.

10. Simulation (Monte Carlo Simulation)
Run 10,000 simulations:
- model stock prices using a random walk model;
- build a distribution of final values;
- visualize 50 random trajectories;
- find the probability of falling below a given level.

11. Creating a Class and Object
- Create a Student class with the fields name, *group*, *GPA*. Implement a method that displays complete information about the student.

12. Encapsulation and getters/setters
Develop a BankAccount class with a private *balance* field. Implement *the deposit()* and *withdraw() methods* to check the correctness of the amount.

13. Using constructors
- Create a Book class with multiple constructors:
- empty
- with the title and author,
- with the title, author and year of publication.

14. Create the classes Transport (base), Car, Bike (descendants). Implement the general *move() method* and override it in the descendants.

15. Create an array of objects of the Animal base class containing Dog, Cat, and Cow objects. Implement the *sound() method*, which is called polymorphically.

16. Create an abstract Shape class with an abstract getArea*() method*. Implement the Circle and Rectangle descendants.

17. Create an ILogger interface with the *log(message) method*. Implement two classes:
ConsoleLogger
FileLoggerwhich handle messages differently.

18. Create a Library class that stores a list of books and provides methods:
- add a book,
- delete the workbook,
- Find a book by title.

19. Implement your own InvalidAgeException, which is thrown when you try to set an age less than 0 or greater than 120.

20. Develop a simplified Online Store system that includes classes:
- Product,
- Customer,
- Order,
- OrderItem.
- Implement:
- adding products to the order;
- calculation of the total cost;
- Display of information about the order.

Director of the Institute
Energy and Digital Technologies _____ **Toygozhinova A.Zh.**

Head of the Department
Information and Communication Technologies _____ **Kassymova D.T.**