1. Nazwa przedmiotu:
Programowanie w języku Java

2. Typ zajęć:
Konwersatorium, 30 godzin (obowiązkowe)
Laboratorium, 30 godzin

3. Koordynatorzy:
Bartosz Zieliński

4. Prowadzący grupy:
Bartosz Zieliński

5. Zaliczenie:
Egzamin

6. Bilans punktów ECTS:
Udział w konwersatoriach - 30 godzin
Udział w zajęciach laboratoryjnych - 30 godzin
Samodzielna implementacja zadań programistycznych – 90 godzin
Przygotowanie do egzaminu, analiza programów do egzaminu oraz udział w egzaminie - 30 godzin
Łączny nakład pracy studenta: 180 godzin , co odpowiada 6 punktom ECTS

7. Efekty kształcenia:
Symbol - Efekt kształcenia - Odniesienie do efektów kierunkowych
- E1 - zna biegle język Java - K_W06+++, K_U04+++
- E2 - potrafi posługiwać się środowiskiem programisty Eclipse lub Netbeans - K_U09++

8. Forma i warunki zaliczenia:
Student otrzymuje ocenę końcową z ćwiczeń na podstawie punktów przyznanych za systematyczne oddawanie zadań wykonywanych w trakcie laboratoriów i zadań domowych. Warunkiem otrzymania zaliczenia ćwiczeń jest uzyskanie łącznie co najmniej 50% możliwych do zdobycia punktów.

Do egzaminu dopuszczaniu są jedynie studenci, którzy uzyskali zaliczenie z laboratoriów. Ocena końcowa z kursu jest wystawiana na podstawie wyniku egzaminu pisemnego, z którego należy uzyskać co najmniej 50% możliwych do zdobycia punktów.

9. Metody sprawdzania i kryteria oceny efektów kształcenia uzyskanych przez studentów:
Programistyczne zadania oddawane w trakcie laboratoriów
Programistyczne zadania domowe
Egzamin

10. Metody dydaktyczne:
Konwersatorium ilustrowane prezentacją komputerową
Ćwiczenia w laboratorium komputerowym, połączone z dyskusją przy tablicy
Samodzielna implementacja zadań programistycznych

11. Wymagania wstępne:
Zdanie egzaminu z kursu Programowanie 2

12. Skrócony opis:
Kurs obejmuje materiał wymagany na egzaminie Java SE 8 Programmer II, przy czym zagadnienia przerabiane na kursie Programowanie 2 będą jedynie uszczegóławiane.

13. Pełen opis:

Java Class Design
- Implement encapsulation
- Implement inheritance including visibility modifiers and composition
- Implement polymorphism
- Override hashCode, equals, and toString methods from Object class
- Create and use singleton classes and immutable classes
- Develop code that uses static keyword on initialize blocks, variables, methods, and classes

Advanced Java Class Design
- Develop code that uses abstract classes and methods
- Develop code that uses final keyword
- Create inner classes including static inner class, local class, nested class, and anonymous inner class
- Use enumerated types including methods, and constructors in an enum type
- Develop code that declares, implements and/or extends interfaces and use the atOverride annotation.
- Create and use Lambda expressions

Generics and Collections
- Create and use a generic class
- Create and use ArrayList, TreeSet, TreeMap, and ArrayDeque objects
- Use java.util.Comparator and java.lang.Comparable interfaces
- Collections Streams and Filters
- Iterate using forEach methods of Streams and List
- Describe Stream interface and Stream pipeline
- Filter a collection by using lambda expressions
- Use method references with Streams

Lambda Built-in Functional Interfaces
- Use the built-in interfaces included in the java.util.function package such as Predicate, Consumer, Function, and Supplier
- Develop code that uses primitive versions of functional interfaces
- Develop code that uses binary versions of functional interfaces
- Develop code that uses the UnaryOperator interface

Java Stream API
- Develop code to extract data from an object using peek() and map() methods including primitive versions of the map() method
- Search for data by using search methods of the Stream classes including findFirst, findAny, anyMatch, allMatch, noneMatch
- Develop code that uses the Optional class
- Develop code that uses Stream data methods and calculation methods
- Sort a collection using Stream API
- Save results to a collection using the collect method and group/partition data using the Collectors class
- Use of () and flatMap() methods of the Stream API

Exceptions and Assertions
- Use try-catch and throw statements
- Use catch, multi-catch, and finally clauses
- Use Autoclose resources with a try-with-resources statement
- Create custom exceptions and Auto-closeable resources
- Test invariants by using assertions

Use Java SE 8 Date/Time API

- Create and manage date-based and time-based events including a combination of date and time into a single object using LocalDate, LocalTime, LocalDateTime, Instant, Period, and Duration
- Work with dates and times across timezones and manage changes resulting from daylight savings including Format date and times values
- Define and create and manage date-based and time-based events using Instant, Period, Duration, and TemporalUnit

Java I/O Fundamentals
- Read and write data from the console
- Use BufferedReader, BufferedWriter, File, FileReader, FileWriter, FileInputStream, FileOutputStream, ObjectOutputStream, ObjectInputStream, and PrintWriter in the java.iopackage.

Java File I/O (NIO.2)
- Use Path interface to operate on file and directory paths
- Use Files class to check, read, delete, copy, move, manage metadata of a file or directory

Use Stream API with NIO.2

Java Concurrency
- Create worker threads using Runnable, Callable and use an ExecutorService to concurrently execute tasks
- Identify potential threading problems among deadlock, starvation, livelock, and race conditions
- Use synchronized keyword and java.util.concurrent.atomic package to control the order of thread execution
- Use java.util.concurrent collections and classes including CyclicBarrier and CopyOnWriteArrayList
- Use parallel Fork/Join Framework
- Use parallel Streams including reduction, decomposition, merging processes, pipelines and performance.

Building Database Applications with JDBC
- Describe the interfaces that make up the core of the JDBC API including the Driver, Connection, Statement, and ResultSet interfaces and their relationship to provider implementations
- Identify the components required to connect to a database using the DriverManager class including the JDBC URL
- Submit queries and read results from the database including creating statements, returning result sets, iterating through the results, and properly closing result sets, statements, and connections

Localization
- Describe the advantages of localizing an application
- Read and set the locale by using the Locale object
- Create and read a Properties file
- Build a resource bundle for each locale and load a resource bundle in an application

14. Literatura:

- Herbert Schildt, Java - the complete reference 9th edition, Oracle Press, 2014
- Kathy Sierra, Bert Bates, OCA/OCP Java SE 7 Programmer I & II study guide, Oracle Press, 2015